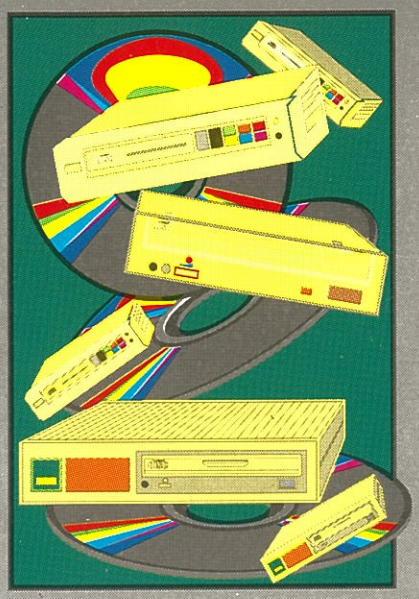
Spooler

MAGAZINE DOS UTILIZADORES PC

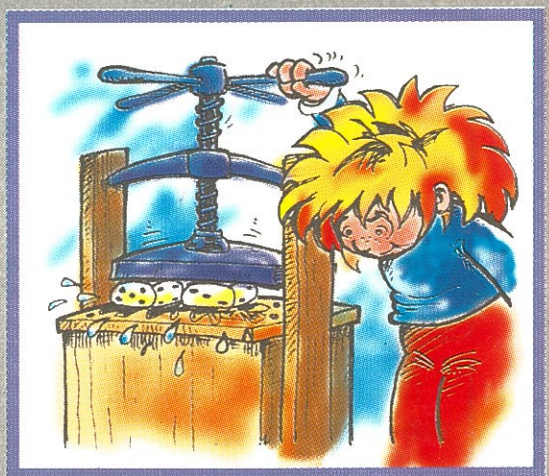
REVISTA MENSAL N.º 28 JANEIRO DE 1994 PREÇO: 800$00 (Cont.)



CD-ROM para PC

JOGO DAS PALAVRAS (I)

MEGA



MÉTODOS DE COMPRESSÃO DE DADOS (I)

- Rotinas (II)
- Atribut
- A Programação Orientada por Objectos
- Super Teste
- SpeedKey
- MIL Fontes
- Impressoras (VI)
- Shareware (II)
- Memória Estendida (IV)
- Etc...

A disqueta que acompanha cada exemplar da revista é parte integrante da mesma e não pode ser comercializada separadamente.

PASSATEMPO SPOOLER «PROGRAMAS CIENTÍFICOS» – Veja no verso de capa como participar

Passatempo Spooler

Programas Científicos

PRÉMIOS

A lista completa de prémios será divulgada na Spooler N.º 29 (Fevereiro 1994). Todos os prémios são patrocinados por prestigiadas empresas do nosso mercado informático que gostosamente aderiram (ou vão aderir) a este projecto. Depois de «fechada» a recepção dos prémios serão constituídos, pelo menos, três «pacotes» correspondentes a outros tantos prémios a atribuir aos melhores programas.

Já confirmados:

Software da extensa gama de produtos representados por:

prológica sistemas informáticos, s.a.

no valor de **150 000$00**

Livros de Informática a escolher entre as obras das editoras FCA e ZIFF-DAVIS PRESS. Uma oferta de:

LIDEL EDIÇÕES TÉCNICAS, LDA.

no valor de **30 000$00**

1 Fax/Modem Int. 14400/14400 bps c/ MNP5, v.42 e V.42 bis oferecido por:

RENAFER EQUIPAMENTOS ELECTRÓNICOS, LDA.

no valor de **76 317$00**

1 Impressora Epson LQ 100 – 24 agulhas oferecida por:

Game® informática

no valor de **55 564$00**

Livros de Informática oferecidos por:

McGraw-Hill Editora McGraw-Hill de Portugal, Lda.

no valor de **50 000$00**

HÁ muito que a comunidade **Spooler** aguardava por uma nova iniciativa no capítulo dos «Passatempos». Chegou a altura de a concretizarmos. Desta vez o tema orienta-se para uma ampla gama de conhecimentos e oferece inúmeras possibilidades à imaginação, à sabedoria e ao bom gosto dos programadores. Não existem limitações: o *software* pode correr tanto em DOS como em Windows, nos modos texto ou gráfico. O carácter didático dos programas será, obviamente, factor de peso na avaliação.

Regulamento

1 – O *software* deverá correr em qualquer PC compatível.

2 – O programa e eventuais rotinas acessórias deverão ser originais e não violar os direitos de *copyright* em vigor em Portugal ou noutros países.

3 – É obrigatório o envio simultâneo de todo o código-fonte.

4 – As principais instruções de utilização deverão ser acessíveis dentro do próprio programa, o que não dispensa o envio de um ficheiro de texto contemplando a instalação, o funcionamento, os aspectos relevantes da programação e os aspectos teóricos da matéria tratada.

5 – É vedada a participação a elementos ligados à **Spooler** ou às empresas patrocinadoras.

6 – Os prémios não poderão ser convertidos em dinheiro.

7 – O material deverá ser entregue nas instalações da **Spooler**, em mão ou por correio registado, até ao dia **31 de Março de 1994**, numa disqueta de 3 1/2".

8 – A apreciação dos programas será feita por um júri composto por colaboradores da **Spooler**.

9 – Os resultados serão divulgados na revista de Maio de 1994.

3 Programas «Cubo2 »+ 3 Programas «Extracto» oferecidos por:

EISA

no valor de **203 754$00**

Spooler
MAGAZINE DOS UTILIZADORES PC

N.º 28 – JANEIRO 1994

UM NOVO ANO

A entrada de um novo ano é sempre um incentivo para concretizarmos ideias acumuladas ao longo dos 365 dias que ficaram para trás. Também para «balanços», «análises de consciência», «autocríticas» e «solenes promessas». A **Spooler** tem consciência do que foi feito em prol da qualidade da revista e daquilo que ainda não foi possível fazer, mas que continua no «saco» dos projectos aguardando condições mais propícias. Não enveredaremos pela facilidade de repetirmos o que temos vindo a anunciar nestas linhas como «conquistas»; nem pela via das promessas. Mas consideramos perfeitamente legítimo transmitir aos nossos leitores a intenção de nos mantermos fiéis às linhas de orientação que têm feito da **Spooler** aquilo que esta é: a única revista de informática publicada em Portugal exclusivamente feita por portugueses e para os portugueses.

UM NOVO VISUAL

Para iniciar o ano, o número de Janeiro apresenta-se com um novo visual. Nada de transcendente, mas reflectindo a nossa preocupação de ir despejando, aos poucos, o tal «saco» de projectos: um novo arranjo gráfico destinado a uma mais fácil leitura.

AS NOVAS SÉRIES

Este número caracteriza-se pelo lançamento de três séries que prometem prender a atenção dos nossos leitores. **Programação Orientada por Objectos** que, como nos diz Paulo Bernardo, «é um tema verdadeiramente quente e que, a ser tratado na **Spooler**, teria de o ser com a habitual seriedade e profundidade que caracterizam as matérias abordadas nestas páginas». Em **Métodos de Compressão de Dados**, José Iria diz-nos «A razão que leva um utilizador a comprimir os seus dados é bastante óbvia: diminuir o espaço ocupado pelos mesmos...». Finalmente, Pedro Andrade na sua nova série **Ordenação e Localização de Dados**, explica-nos que «A ordenação consiste em transformar um conjunto de dados desordenados numa lista com uma ordem lógica...».

Renato Casquilho

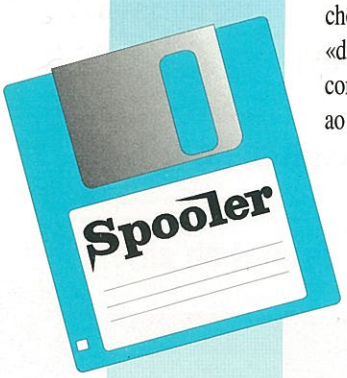
Spooler

- **Propriedade:** Graficria, Publicidade e Artes Gráficas, Lda.
- **Director de Edição:** Franco Gomes ■ **Director**: Renato Casquilho
- **Redacção:** Carlos Ladeira e Renato Casquilho.
- **Colaboradores:** Afonso Infante, Eduardo Cardoso, José Orlando Pereira, José Páscoa, Miguel Vitorino, Paulo Bernardo.
- **Ilustrações:** Leonor Gomes.
- **Relações Públicas e Publicidade:** Fátima Ribeiro.
- **Assinaturas e encomendas:** Guiomar Russo e Paula Martinez.
- **Redacção, Publicidade e Serviços Administrativos**: Rua Duarte Pacheco Pereira, 69-A – Damaia – 2700 AMADORA – Telfs.: 475 22 89/474 60 11/41
- **Produção Gráfica:** Graficria, Publicidade e Artes Gráficas, Lda.
- **Impressão:** Empresa Litográfica do Sul, S.A. ■ **Distribuição:** Electroliber
- **Revista mensal** ■ **Preço:** 800$00 ■ **SRIP N.º** 213900
- **Tiragem:** 12 500 exemplares ■ **Dep. Legal N.º:** 28587/89

2 O Conteúdo da Disqueta
3 Jogo das Palavras (Parte I)
4 Rotinas (Parte II)
5 Mega-Comm
9 Atribut
10 A Programação Orientada por Objectos
13 Fontes
14 Super Teste
16 SpeedKey
17 O Fantástico Mundo das BBS (VI)
20 Métodos de Compressão de Dados (I)
23 Ordenação e Localização de Dados
26 MIL Fontes
28 Impressoras (VI)
31 Shareware (II)
33 Memória Estendida (IV)
37 A Revolução do Multimedia (II)
40 Livros

EM SUPLEMENTO
MONTRA SPOOLER

O CONTEÚDO DA



ATRIBUTO.EXE

Conforme prometido, aqui estão os restantes utilitários do pacote «File Manager» da autoria de António Costa e Fernando Pinheiro, cujos dois primeiros foram publicados o mês passado.

COMPRES1.EXE

«Saudações a todos os programadores (amadores e profissionais) e gentes ligadas à Informática. Tenho 14 anos e programo em Turbo Pascal há cerca de dois anos, podendo-se dizer que domino (bastante) bem a «fera». Também me estou a iniciar na programação em C, que estou a achar bastante interessante». Estas são as frases introdutórias com que José Iria dá início a uma série que promete ser bastante interessante: «Métodos de Compressão de Dados».

FONTES.EXE

Nunca é de mais insistir-se em temas que consideramos importantes. Esta matéria, se bem que não da mesma forma, já foi abordada o mês passado. Mas António Mendoça acrescenta-lhe a utilização do «Sopa de Letras» de Armando Frazão, que tanto sucesso tem tido entre a comunidade **Spooler**.

FREE-S28.COM

Graças ao utilitário concebido pelo nosso colaborador Carlos Ladeira (ver o artigo «TXT2COM») da **Spooler Nº 27**, a listagem do *Shareware* à disposição na «Montra Spooler» passa a ser «visível» sem o auxílio de qualquer editor de texto. Basta digitar FREE-S28 e o ecrã ficará inundado de informação. E até pode imprimir... Neste número temos mais 8 (oito) referências novas!

JPALAVRA.EXE

Ora aqui temos um joguinho mesmo a calhar para os «sabichões» dos concursos da TV. Escrito em Pascal, contém muitas «dicas» para os fãs desta linguagem. O Nuno Morgado promete continuar no próximo número, acrescentando novas capacidades ao seu jogo.

JUMPER_.EXE

Para os amadores de puzzles, aqui vai um interessante «quebra-cabeças» *shareware* que a **Spooler** põe à disposição dos seus leitores. Atenção: só corre em Windows!

MEGACOM.EXE

Como estamos em «maré» de BBS e comunicações, considerámos interessante oferecer este programa aos nossos estimados leitores. Não é nenhuma «espada», mas experimentem e depois digam-nos as vossas impressões. O Pedro Duarte esmerou-se na escrita, pelo que não deverão ter dificuldades. Se tiverem modem, claro...

ORDENA1.EXE

«Duas das principais funções relacionadas com o armazenamento de dados são a ordenação e a localização. Neste artigo trataremos a ordenação, ficando a localização para depois». Pedro Andrade, já nosso conhecido da série «Programação em Modo Gráfico para Turbo Pascal», introduz-nos, com estas palavras, numa nova e aliciante série que hoje se inicia.

PRIN_FON.EXE

Miguel Vitorino oferece-nos mais um dos seus interessantes utilitários para Windows. O artigo, a não perder, começa assim: «Não, não lhe vamos oferecer mil fontes para o seu Windows! Ao invés, vamos-lhe oferecer na disqueta **Spooler** deste mês algo muito mais útil: um programa para listar todas as suas fontes».

ROTINAS.EXE

Cá temos de novo o António José com mais um conjunto de rotinas para Pascal. O seu artigo contém todas as explicações.

SPEED_K.EXE

Este é um dos tais programas «condenados» a ir parar ao nosso directório de utilitários. Já está nos nossos computadores e realmente é «útil». O Tiago Santos explica-nos a sua filosofia.

TESTE.EXE

Não vão faltar, certamente, leitores interessados neste curioso programa do nosso Miguel Gonçalves. Curiosos (se surpreendentes...) também os resultados obtidos com o exemplo que o autor incluiu. Esperamos que se divirtam e não fiquem complexados...

TOWERS_.EXE

Mais um explêndido *shareware* para Windows. Trata-se de um jogo de cartas, tipo «solitário», mas muito, muito mais difícil...

JOGO DAS PALAVRAS

PARTE I

colaboração do leitor

Ora aqui temos um joguinho mesmo a calhar para os «sabichões» dos concursos da TV. Escrito em Pascal, contém muitas «dicas» para os fãs desta linguagem. O Nuno Morgado promete continuar no próximo número, acrescentando novas capacidades ao seu jogo.

A ideia deste jogo foi tirada de um livro de programas e jogos para o SPECTRUM.

Para «correr» o jogo tem de se escrever na linha de comando JP e **C** conforme tenha uma placa EGA/VGA ou **M** caso contrário.

É um jogo muito simples: dada uma palavra (quer por um amigo ou quer pelo computador) é preciso descobri-la (esta é a parte difícil) e voltar a pô-la como na forma original (coisa não muito difícil depois de vários jogos), uma vez que ela foi baralhada.

Se o adversário escolhido for humano, então terá de escrever uma palavra sem que nós vejamos (convém, por uma questão de dificuldade, começar todas as palavras com letra maiúscula). Se o adversário for o computador, ele escolherá uma palavra «ao calhas» das existentes no ficheiro (PALAVRAS.JP). Depois de escolher o nível de dificuldade o computador baralhará a palavra.

Durante o jogo aparece a informação necessária para se jogar, apenas valendo a pena dizer que quando se escolhe «FIM» na ordem começa-se pelas primeiras letras e quando se escolhe «PRINCÍPIO» começa-se pelas últimas. Convém ainda dizer que, se obteve uma pontuação digna das melhores classificações (menor Nº de jogadas feitas e em caso de empate ganha o que tiver a maior palavra), poderá escrever o seu nome e visioná-lo. Caso não tenha obtido a pontuação necessária, voltará imediatamente para o menu principal.

Podem apagar-se os ficheiros de informação do jogo (PALAVRAS.JP e HISCORES.JP). Isto serve, por exemplo, para apagar todas as palavras existentes em disco e apagar as melhores classificações existentes, respectivamente. Claro que quando for jogar na opção contra o computador, este o avisará que será preciso introduzir palavras (que poderá fazer com o INSP descrito no próximo número). Há ainda a opção de retirar todos os sons premindo-se **S** no ecrã de entrada (esta opção não anula os *beeps* que podem acontecer antes deste).

Convém referir que o jogo será um pouco mais rápido sem sons (enquanto está a dar um som, o computador não faz mais nada) mas também isso tem pouco interesse para o jogo em questão.

Por questões de espaço não é possível neste número incluir o INSP (Inserir Palavras), o MOSTRAP (Mostrar Palavras), o APAGARP (Apagar Palavras) e o MODIFP (Modificar Palavras). Serão publicados no próximo número juntamente com o artigo a eles referente.

Em princípio este jogo funciona em qualquer computador (por ser em modo de texto), mas algumas palavras poderão não se ver. Se isto acontecer contactem-me através da **Spooler**. Aviso que não funciona em 40x25 linhas (a maioria (senão todos!) trabalham em 80x25 linhas).

Vai também junto o JP.ICO, um ícone para os utilizadores do

WINDOWS que está um bocado «tosco» mas serve para improvisar nos utilizadores que não dispuserem de um «desenhador» de ícones.

NOTA: Se ao sair do jogo aparecer uma mensagem de *Write Protect* (disqueta protegida para escrita) seleccionem o **A** para abortar. Este erro é devido a se ter seleccionado não gravar se a disqueta estiver protegida.

Quanto ao jogo é tudo. Divirtam-se!

AGORA QUANTO À PROGRAMAÇÃO:

Este jogo foi programado em Turbo Pascal (inicialmente em 5.5 passando pelo 6.0 e acabando no 7.0). Todos os executáveis foram compilados no T.P. 7.0. Dado que comecei a fazer o jogo em 1991 (como fiz o jogo por diversão não tive pressa em o acabar) e nessa altura ainda não sabia o que sei agora, agravando-se pelo meu método de programar (de improviso), o código do jogo está ficou pouco confuso. Como fiz sem grandes cuidados na legibilidade do código, as partes mais confusas já melhorei um pouco (tirei bastantes *labels*, variáveis cujo nome não estava de acordo com a função desempenhada, etc.) mas ainda há algumas coisas confusas (montes de variáveis globais, em vez de passagem de valores às funções utilização das variáveis globais, etc.).

A base do programa assenta no **BaralharPalavra** e no **ActualizarArray**, que baralham as palavras e vão actualizando as palavras durante o jogo respectivamente. O BaralharPalavra utiliza o ActualizarArray para baralhar a palavra. Faz o seguinte: com um ciclo FOR com o Nº de vezes que se baralha a palavra, ele escolhe uma letra «ao calhas» da palavra e mete a letra uma vez para o fim, outra vez para o princípio. O ActualizarArray recebe o Nº da letra na palavra (a contar do princípio), vai ao *array* buscá-la e guarda-a numa variável à parte. Depois é só copiar a letra a seguir para a posição actual, avançar e repetir até que se esteja na penúltima posição se a ordem for para o fim e ao contrário se a ordem for para o princípio. A letra guardada na variável vai para o fim ou o princípio conforme a ordem. Ainda utilizo parâmetros com o PARAMSTR e o tempo de utilização com o GETTIME. O JP.EXE tem ainda a particularidade de conter o EGAVGA.BGI (ficheiro do modo gráfico de EGA e VGA) e todos os CHR (fontes no modo gráfico) incorporados.

Nuno André Porto Morgado

> **A ideia deste jogo foi tirada de um livro de programas e jogos para o SPECTRUM.**

colaboração do leitor

ROTINAS

PARTE II

Cá temos de novo o António José com mais um conjunto de rotinas para Pascal.

Cá estou eu, mais uma vez, com novas rotinas escritas em Pascal. No fundo, trata-se da continuação das rotinas anteriormente publicadas. Por isso mesmo, intitulei este conjunto de rotinas de «Rotinas II».

Desta vez, criei 9 rotinas bastante úteis e, de seguida, vou explicar, com exemplos, algumas dessas rotinas:

> **Desta vez, criei 9 rotinas bastante úteis...**

- **Ponto:** Separa um determinado valor com pontos de 3 em 3 dígitos. Tal torna-se mais cómodo e mais apresentável.

Biblioteca(s) utilizada(s): Nenhuma

Modo de utilização: Ponto(N:Longint)

Exemplo: Writeln(Ponto(123456)+' Bytes'));
Escreve no ecrã o seguinte: 123.456 Bytes

- **Espacejamento:** Permite a selecção do espacejamento entre as linhas imprimidas.

As escolhas possíveis são as seguintes:

- 1 : Espacejamento de 1/6 polegadas.
- 2 : Espacejamento de 1/8 polegadas.
- 3 : Espacejamento de 7/72 polegadas.

Biblioteca(s) utilizada(s): Printer

Modo de utilização: Espacejamento(E:Integer)

Exemplo: Espacejamento(3);
Selecciona um espacejamento de 7/72 polegadas entre cada linha.

– Rotinas de páginas: Essas rotinas permitem visualizar páginas de ecrã de forma mais rápida e, sem que seja necessário esperar que um menu seja desenhado ou outra coisa qualquer. Para tal, são necessários três procedimentos:

- **Inicia_Pagina** (Inicialização das páginas);
- **Salva_Pagina** (Grava as páginas);
- **Repoe_Pagina** (Põe no ecrã as páginas).

Os dois últimos procedimentos necessitam de um parâmetro que corresponderá ao Nº das páginas.

Biblioteca(s) utilizada(s): Nenhuma

Modo de utilização: Inicia_Pagina
Salva_Pagina(Pagina:Byte)
Repoe_Pagina(Pagina:Byte)

Exemplos:

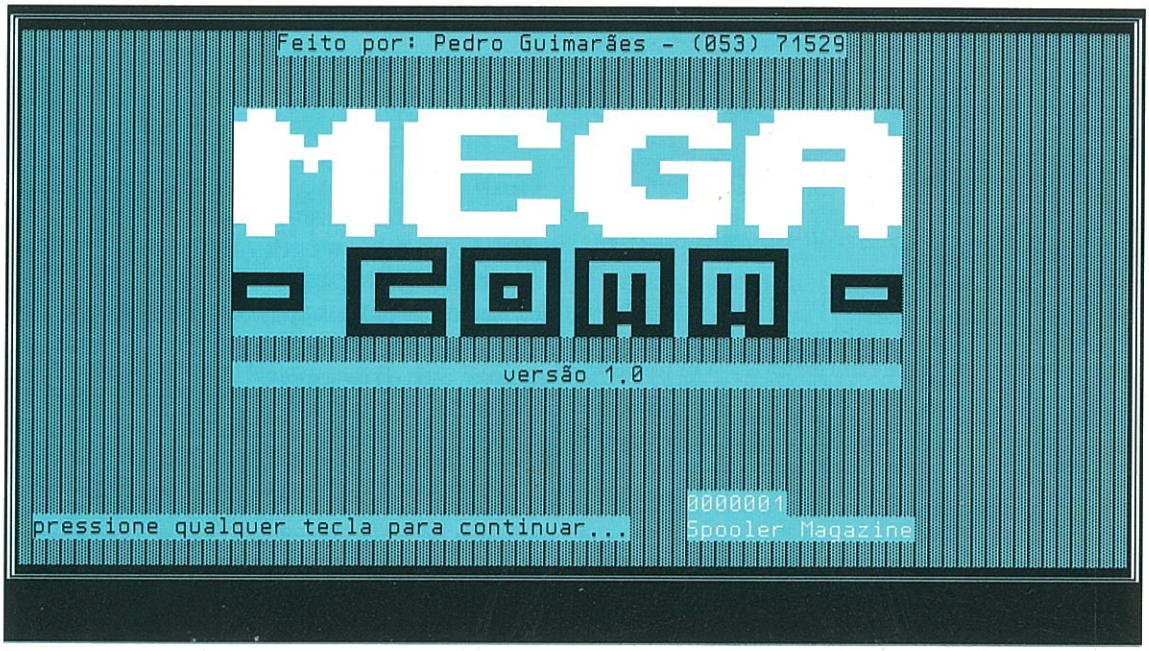
```
Inicia_Pagina;
        Clrscr;
        For I:=1 To 24 Do
                Writeln('Spooler Magazine');
        Salva_Pagina(1);
        Clrscr;
        For I:=1 To 24 Do
                Writeln('ABC de Braga');
        Salva_Pagina(2);
        Clrscr;
        Repeat
                Repoe_Pagina(1);
                A:=Lendo_Tecla;
                Repoe_Pagina(2);
                A:=Lendo_Tecla
        Until A=Chr(27);
        ...
```

Este exemplo permite trocar páginas rapidamente até que seja premida a tecla ESC.

António José Monteiro Pereira

MEGA

colaboração do leitor



Como estamos em «maré» de BBS e comunicações, considerámos interessante oferecer este programa aos nossos estimados leitores. Não é nenhuma «espada», mas experimentem e depois digam-nos as vossas impressões. O Pedro Duarte esmerou-se na escrita, pelo que não deverão ter dificuldades. Se tiverem modem, claro...

Este programa não é suposto ser um programa de «concorrência», ou seja, não foi feito para ser usado em vez do programa de comunicações a que estamos habituados. A principal função deste programa é ajudar programadores iniciados com ideias para os seus programas. Para isso, junto a sua fonte, que deverá ser corrida com o Microsoft QuickBasic 4.5.

USO E FUNÇÃO DOS FICHEIROS EDITÁVEIS DE CONFIGURAÇÃO

Os ficheiros:

AGENDA.FON	(números de telefone e respectivos destinos)
PROTOCOL.MCM	(protocolos externos)
MACROS.MCM	(Macros)
EDITOR.BAT	(File batch de acesso a editor externo)
SETUP.BBS e SETUP.INT	(Ficheiros de configuração interna)

têm uma função e tamanho específico. Além disso estão todos em ASCII e podem ser editados. Todos podem conter um número máximo de linhas.

AGENDA.FON – É o ficheiro onde se escreve o nome de um sistema remoto e o seu número de telefone. Pode ter até 100 nomes, ou seja, 200 linhas, uma com o nome e outra com o número, e assim sucessivamente num limite de 100.

Exemplo:

Ficheiro «Agenda.fon»

```
até x 100   Mega BBS      (nome)
            053-71529     (número)
            X-User BBS
            02-234543
            .
            .
```

PROTOCOL.MCM – Só pode até 42 linhas, ou seja, uma linha com o nome do protocolo, outra com o comando DOS de Dload e outra com o comando DOS de Upload, tudo isto vezes 14. Caso necessite de incluir a velocidade, porta COM ou nome do ficheiro no comando, faça-o usando os substituintes **S**, **C** e **F**, respectivamente.

Exemplo:

Ficheiro «Protocol.mcm»

```
até x 14    ZMODEM batch          (nome)
            c:\mega\z\zmodem.exe | c ls rz     (comando p/ dload)
            c:\mega\z\zmodem.exe | c ls sz lf (comando p/ upload)
            XMODEM 1-K
            c:\mega\x\xmodem.exe | c ls
            c:\mega\x\xmodem.exe | c ls send lf
            .
            .
```

> **A principal função deste programa é ajudar programadores iniciados com ideias para os seus programas.**

MACROS.MCM – Só pode ter até 10 linhas, ou seja, cada linha contendo o macro.

Pode incluir o carácter RETURN (ASCII 13), o carácter ESCAPE (ASCII 27), a data e as horas através dos substituintes **r**, **e**, **d** e **t** respectivamente. Para uma melhor elucidação consulte a explicação sobre a função MACROS.

Exemplo:

Ficheiro «Macros.mcm»

```
até x 10      Hallo world !
              // Pedro Duarte \\
              Origin: Lusitânia BBS - Braga
              .
              .
```

EDITOR.BAT – Ficheiro *batch* que dá acesso ao editor externo. Pode ser editado, e deve ser composto por:

Exemplo:

Ficheiro «Editor.bat»

```
@ECHO OFF
[nome do editor] %1
CLS
[nome do editor] – referente ao ficheiro
executável do editor e, se necessário, com o
respectivo caminho (path).
%1 – não deve ser excluído este termo, pois
substitui
o nome do ficheiro a editar definido no programa
MegaCOMM.
```

SETUP.BBS e SETUP.INT – Estes ficheiros, para não haver complicações, apenas devem ser alterados com o utilitário SETUP.EXE.

Atenção: Os únicos ficheiros que devem ser editados são o AGENDA.FON, PROTOCOL.MCM, MACROS.MCM e EDITOR.BAT.

FUNÇÕES DO PROGRAMA

O programa MegaCOMM trabalha de forma simples e idêntica aos programas que usam menus de janelas, cursor, PgUp, PgDn, Enter, e Esc. No entanto aqui fica uma breve explicação sobre as funções da MegaCOMM :

Função AGENDA:

Uma vez escolhida esta função, uma janela lhe aparecerá, onde estarão visíveis os vários nomes de pessoas e sistemas para onde pode telefonar que estão definidos no ficheiro «AGENDA.FON».

Função MARCAR NÚMERO:

Esta função tem um funcionamento idêntico à anterior, mas desta vez é o próprio utilizador a introduzir o número.

Função ECHO:

Tendo esta função ligada, qualquer tecla pressionada será ecoada no ecrã, isto é, mesmo que o sistema com o qual você está ligado não lhe «devolva» a tecla que pressionou, esta será na mesma enviada para o ecrã. Caso o sistema remoto «devolva» a tecla pressionada, então esta aparecerá duas vezes no ecrã. Caso esta função já tenha sido anteriormente seleccionada, e a voltou a seleccionar, então ela desligar-se-á.

Função CAPTURA:

Escolhendo esta opção, um nome de um ficheiro ser-lhe--á solicitado e nele serão «depositadas» todas as informações que aparecem no ecrã. Caso volte a escolher esta função, já a tendo escolhido anteriormente, a capturação dar-se-á por concluída. O ficheiro onde ficará depositada a captura pode ser visualizado com um simples editor de texto ASCII ou com o comando «type» do DOS. Os comandos ANSI serão filtrados, não aparecendo, por isso, no ficheiro de captura.

Nota: Nunca defina um ficheiro de captura (ou fotografia – ver mais à frente) como tendo nome idêntico a um dos ficheiros do MegaCOMM, pois poder-se-á ver em apuros...

Funções DOWNLOAD E UPLOAD:

Com estas funções pode transferir ficheiros com o sistema a que está conectado. Terá também de escolher o protocolo de transferência definido por si no ficheiro «PROTOCOLOS.MCM» e o ficheiro a transferir, caso necessário. *Download* significa capturar ficheiro e *Upload* enviar. Apenas o protocolo ASCII é suportado internamente pelo programa, necessitando todos os outros de serem suportados por ficheiros externos, que se podem encontrar em qualquer BBS.

Função MODO TERMINAL:

Através desta função pode escolher o seu modo de terminal:

1 – ANSI (gráficos a cores);
2 – ASCII (texto monocromático).

Função EDITOR:

Após seleccionar esta função, um nome ser-lhe-á solicitado, que se refere ao ficheiro a editar. Para isso deve ter o ficheiro *batch* editável «EDITOR.BAT» bem construído e de acordo com o seu editor externo de texto.

Função MACROS:

Macros não são mais que frases de escrita imediata e pré-memorizadas. O MegaCOMM permite até 10 Macros definidos no ficheiro «MACROS.MCM». Caso o ficheiro de macros exista, uma janela aparecer-lhe-á com os macros referidos no respectivo ficheiro. Uma vez que deseje introduzir nos seus macros elementos não possíveis através do teclado, pode fazê-lo pelos seguintes substituintes:

r – representa a tecla RETURN
e – representa o código ESCAPE (27)
t – representa as horas do sistema
d – representa a data do sistema

Assim, se por exemplo, desejar incluir um comando em ANSI no seu macro, pode observar a seguinte exemplificação, que deverá ser incluída numa das linhas do ficheiro «macros.mcm»:

```
e[32m MegaCOMM ld lt le[0m lr
```

este macro faz com que o texto fique a verde (e[32m), escreva o texto «MegaCOMM », insira a data (d), insira as horas (t), coloque o texto a normal (e[0m) e finalmente que mude de linha (r).

Funções VER FICHEIROS, MUDAR DIRECTORIA E MUDAR DRIVE:

Estas funções têm um funcionamento idêntico ao do DOS. A primeira resulta numa listagem dos ficheiros da actual directoria, a segunda e a terceira, após a solicitação de um caminho (*path*), mudam de directoria e *drive*, respectivamente.

Função VOLTAR AO TERMINAL OU «ESC»:

Esta função faz com que o programa volte ao modo de terminal.

Função FILES:

Tendo o ficheiro «EDITOR.BAT» em condições de utilização, poderá, através desta opção, redefinir os ficheiros de Macros, Agenda e Protocolos.

Área «COMANDOS»:

Modo AUTO-ATENDIMENTO:

Este modo faz com que o seu computador atenda automaticamente qualquer chamada feita para si por outro modem. No entanto ainda pode fazer chamadas.

Modo CHAMADA:

Este modo «liga» o seu computador apenas para efectuar chamadas e nunca para as receber automaticamente.

Modo ATENDER:

Caso alguém esteja a tentar contactá-lo com um modem pode atender a sua chamada de imediato escolhendo o modo atender, mesmo em modo chamada.

Modo DESLIGAR:

Caso esteja no meio de uma comunicação e deseje acabá--la, escolha esta opção e as comunicações serão terminadas e o programa MegaCOMM reinicializado, fechando o ficheiro de captura (caso esteja aberto) e restaurando todas as outras opções segundo os modos de defeito e de configuração. Tenha atenção às definições feitas na área de períodos de contagem e taxas (COMANDOS/PERÍODOS), que se restaurarão após ter usado esta opção.

Modo DESENCRAVAR:

Existe um comando incluído no menu principal e na opção «comandos», que se chama «DESENCRAVAR». Pois é! Remédio para todos os males. A técnica que usei para criar este comando foi descoberta um bocado ao acaso. Talvez já lhe tenha acontecido alguém levantar o telefone durante a comunicação, ou de repente aparecer muito lixo vindo sabe Deus de onde, etc! Caso costume usar programas como o Telemate, QModem, etc, talvez já tenha experimentado enviar um *BREAK SIGNAL* que de nada resultou... e depois fez o *reset* ao computador, voltou a chamar o programa de comunicações e... *Voilá*! Como a linha não se tinha perdido, a comunicação voltou ao normal. Bem, a função «DESENCRAVAR» funciona um bocado do mesmo modo, mas suprime aquela trabalheira toda de *reset* e etc... apenas desliga e volta a ligar a porta de *comms*, o que me tem resultado sempre, por incrível que pareça!

Existe um comando incluído no menu principal e na opção «comandos», que se chama «DESENCRAVAR». Pois é! Remédio para todos os males.

Modo CHAT:

Caso esteja conectado a um seu amigo ou a um sistema que não possua um sistema de *Chat* desenvolvido, pode sempre ligar esta opção. Assim, tudo o que você escrever ficará registado a vermelho, e o que receber a verde. *Chat* significa conversa, diálogo (quando não é monólogo...).

Fotografia:

Se desejar capturar para um ficheiro o que está de momento a visualizar no ecrã, basta escolher esta subopção e definir o ficheiro onde ficará «depositada» a fotografia. Esta operação demora poucos segundos (conforme o computador) e o ficheiro resultante pode ser visualizado em modo de texto, com (por exemplo) o comando «type» do DOS.

Períodos:

Caso deseje contabilizar as suas chamadas, então escolha esta opção, onde poderá definir o tempo que demora se efectuar 1 período (em segundos) e depois o seu respectivo preço (em escudos). Para mais informações veja mais abaixo a *hotkey* Alt-i, que entre outras coisas lhe permite visualizar o actual gasto desde o início da ligação.

Função AJUDA:

Uma vez seleccionada esta função poderá visualizar este mesmo artigo e tirar uma eventual dúvida.

Funções SHELL E SAIR:

Estas funções provocam um acesso ao DOS, embora a primeira permita voltar ao MegaCOMM no seu estado anterior escrevendo «EXIT» no DOS, e a segunda faça com que a saída ao DOS seja definitiva.

AS «HOT KEYS»:

As *hot keys*, ou teclas de acesso rápido, apenas podem ser usadas no modo terminal e não dentro do menu principal. São elas as seguintes:

Alt-i – Esta tecla dá acesso a uma janela onde aparecerão informações sobre o modem, modos ECHO, ANSI, tempo de conecção e gastos monetários e um relatório sobre todas a Hot Keys. Em relação à função de gastos e tempos (muito útil, pois permite contabilizar chamadas de longa duração ou para grandes distâncias), esta é efectuada da seguinte maneira: sempre que detectar no modo terminal a palavra «CONNECT», inicia a contagem de tempo e gastos monetários e, se detectar a frase «NO CARRIER», desliga a contagem, embora não leve os valores a zero. A palavra «CONNECT» significa que o modem entrou em contacto com outro, e a frase «NO CARRIER», que o contacto foi terminado ou não foi efectuado. Caso inicie uma nova ligação ainda sem ter abandonado o MegaCOMM, os valores da chamada anterior serão ignorados e uma nova contagem se efectuará. Para simular esta função, escreva «CONNECT» e verifique o resultado pressionando ALT-i. Depois escreva «NO CARRIER» e volte a repetir.

Alt-q – Esta tecla permite-lhe sair ao DOS. De preferência nunca a utilize (hehehe!).

Alt-m – Esta tecla permite-lhe aceder ao menu principal, onde apenas poderá escolher as opções com o cursor e enter.

Alt-a – Esta tecla permite-lhe aceder à janela onde estão todos os sistemas e os respectivos números escolhidos por si no ficheiro «AGENDA.FON».

Alt-d – Esta tecla permite-lhe o acesso ao menu de Download (Captura de ficheiros).

Alt-u – Esta tecla permite-lhe o acesso ao menu de Upload (Emissão de ficheiros).

Alt-c – Esta tecla permite-lhe ligar ou desligar o modo CHAT (conversa).

Alt-h – Esta tecla permite-lhe desligar a sua actual comunicação (veja melhor na explicação sobre a função COMANDOS/DESLIGAR).

Alt-s – Esta tecla permite-lhe fazer um Shell ao DOS. Para voltar ao MegaCOMM basta escrever «EXIT» no prompt do DOS.

Alt-w – Esta tecla permite-lhe apagar o ecrã, em caso de excesso de «lixo».

Alt-r – Esta tecla permite-lhe renovar os valores do contador de tempos e gastos monetários. Apenas é possível dentro da janela activada por Alt-i.

Alt-b – Esta tecla permite-lhe desencravar uma eventual «queda de lixo» ou qualquer encravanço. Verifique melhor o uso deste comando na explicação da função COMANDOS/DESENCRAVAR.

Alt-f – Esta tecla permite-lhe tirar uma «fotografia» do actual ecrã. Veja melhor o uso deste comando na explicação da função COMANDOS/FOTOGRAFIA.

Alt-x – Esta tecla permite-lhe visualizar este artigo, para o esclarecimento de eventuais dúvidas.

F1 a F10 – Estas teclas permitem-lhe introduzir na sua comunicação os Macros definidos por si no ficheiro «MACROS.MCM».

> **As *hot keys*, ou teclas de acesso rápido, apenas podem ser usadas no modo terminal e não dentro do menu principal.**

MENSAGENS DE ERRO

Várias mensagens de erro podem aparecer caso se verifiquem determinadas situações, que serão indicadas pelo programa. Caso não haja referência a nenhum erro concreto, é porque se trata possivelmente de um erro na configuração com o utilitário SETUP.EXE, a falta de algum ficheiro ou mesmo problemas «maquinais». Para a primeira situação, o melhor é correr novamente o utilitário SETUP.EXE e definir correctamente a configuração. Para a segunda situação, o melhor será obter uma versão completa do MegaCOMM. Se não consegue resolver o problema, agradecia que me contactasse e que me descrevesse o sucedido. Do mesmo modo agradecia que me fosse comunicada a existência de *Bugs* e/ou sugestões para novas funções. Nesse caso, terei o maior gosto de oferecer, a quem me contactar nesse âmbito, (ou noutro, desde que devidamente justificado) futuras versões do MegaCOMM!

O UTILITÁRIO SETUP.EXE

O utilitário SETUP.EXE destina-se a configurar o programa MegaCOMM conforme o seu sistema. Para isso basta executá-lo e seleccionar as opções devidamente.

Através deste utilitário pode definir vários pontos de configuração do programa MegaCOMM, como a porta COM, Velocidade, Paridade, Data Bits, Stop Bits, *Buffers* de emissão e recepção, *Flow control* e respectivo tamanho de *buffer*, Modo de *LineFeed*, Número de linhas para visualizar, Init String e modo de chamada (Tons ou Impulsos).

Até ao próximo programa, porque este não será o último!

Pedro Duarte Guimarães

ATRIBUT

colaboração do leitor

Conforme prometido, aqui estão os restantes utilitários do pacote «File Manager», cujos dois primeiros foram publicados o mês passado.

Este utilitário faz o mesmo que o ATTRIB do DOS, no entanto é um bom exemplo de como se pode conseguir o mesmo numa linguagem como o PASCAL. Ao programa só devem ser passados no máximo dois parâmetros, sendo eles:

A máscara do(s) ficheiro(s) ao(s) qual/quais deve(m) ser alterados os atributos, e um parâmetro opcional que deve ser indicado no caso de se pretender apenas visualizar os atributos dos ficheiros sem os alterar.

Programação: Este programa utiliza a função do Pascal SETFATTR que serve justamente para alterar os atributos de um ficheiro. Para localizar o nome dos ficheiros a alterar e para saber os seus atributos foram utilizadas novamente as funções FINDFIRST e FINDNEXT.

Uso: ATRIBUT máscara [/V]
/V especifica que os atributos devem ser apenas visualizados.

COMPARA

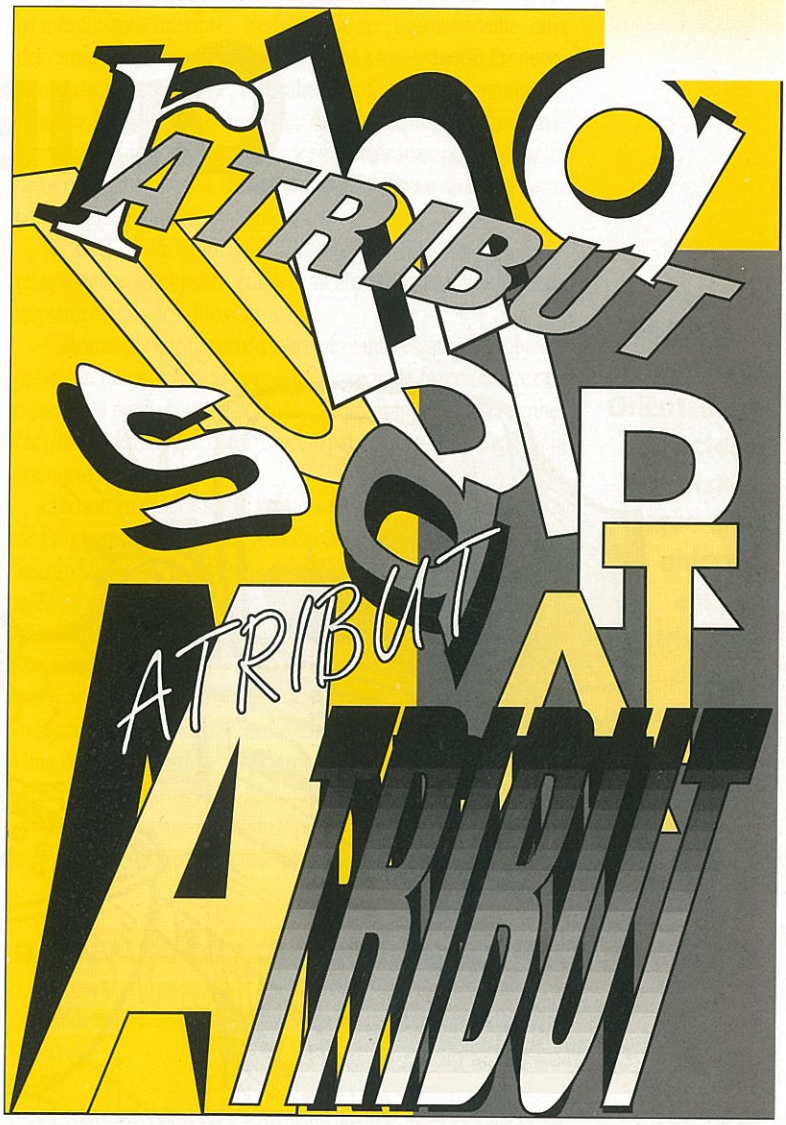
Este é mais um utilitário já existente no DOS e também é destinado unicamente a demonstrar a sua programação. Como já devem ter percebido a sua única função é comparar dois ficheiros e dizer-nos se eles são iguais. Embora um pouco mais pobre que a versão DOS, pois não nos indica os deslocamentos das diferenças, não deixa de ser um bom exemplo de programação.

Programação: Nada de novo a dizer também, se já leu este documento desde o principio há que dizer que ele utiliza as mesmas funções BLOCKREAD e BLOCKWRITE que os programas gêmeos CRIPT/DESCRIPT.

Uso: COMPARA ficheiro1 ficheiro2

CRIPT-DESCRIPT

Este conjunto de dois utilitários tem com objectivo «criptar» e «Descriptar» ficheiros que queiramos ocultar de olhos indesejáveis. Em princípio os ficheiros foram idealizados para «criptar» ficheiros de texto, no entanto nada no programa impede de criptar até ficheiros executáveis.



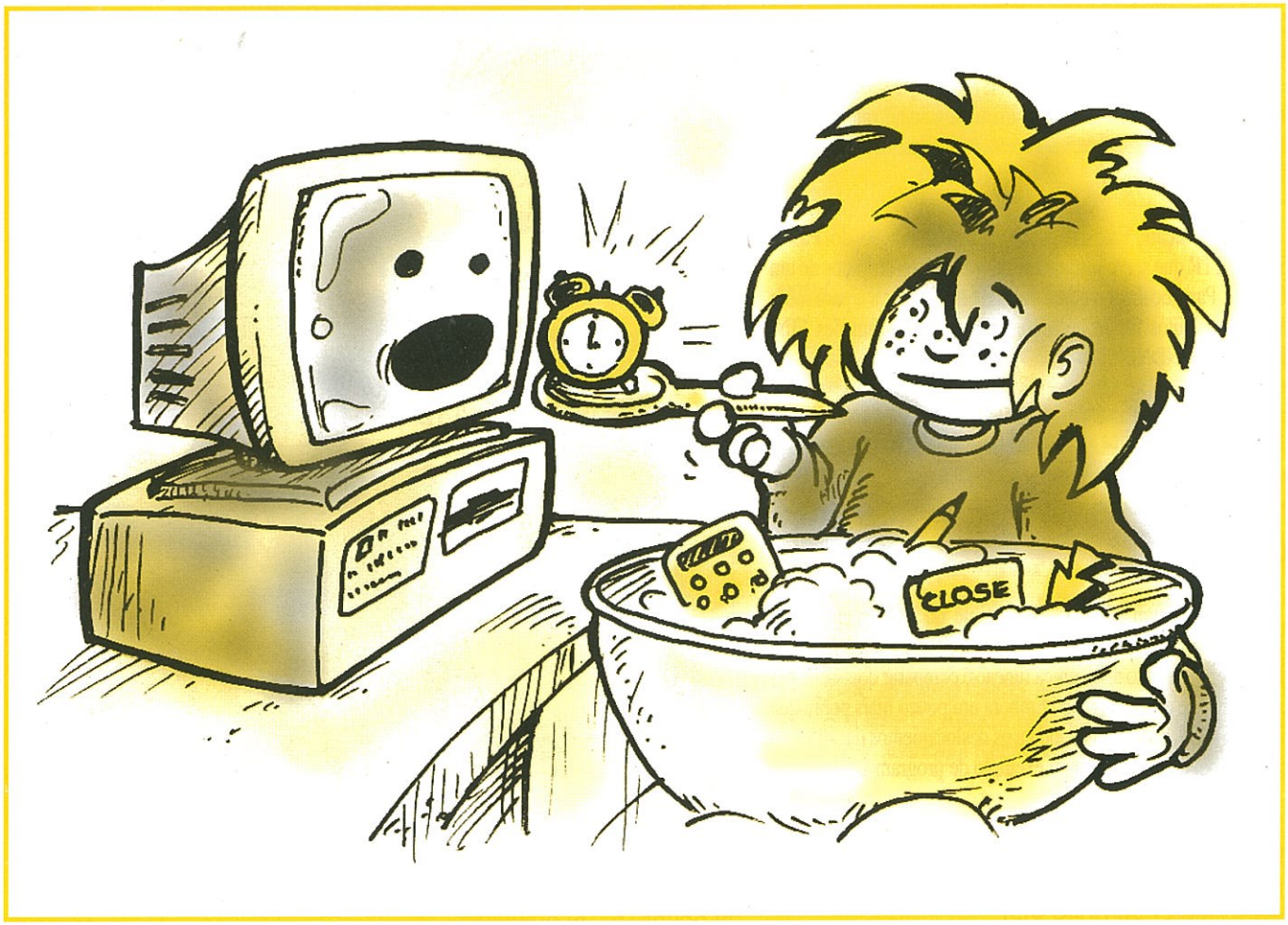
Programação: Estes utilitários utilizam os Procedimentos BLOCKREAD e BLOCKWRITE que podem fazer a leitura e escrita de um número avultado de bytes. Estes mesmos procedimentos utilizam como BUFFER um *array* no programa.

Uso: (DES) CRIPT fx_origem [fx_destino] password [/D]
/D apaga o ficheiro de origem caso seja escolhido (des)criptar para um segundo ficheiro diferente do primeiro.

Fernando Pinheiro & Antonio Costa

PARTE I

A PROGRAMAÇÃO ORIENTADA POR OBJECTOS



Olá leitores.

Após insistentes pedidos, dos quatro cantos do mundo, eis que se escreve (finalmente) na **Spooler** sobre «Programação Orientada por Objectos». Este é um tema verdadeiramente «quente» e que, a ser tratado na **Spooler**, teria de o ser com a habitual seriedade e profundidade que caracterizam as matérias abordadas nestas páginas.

Neste sentido, mais do que elaborar apenas um pequeno texto definindo o conceito de «Programação Orientada por Objectos» de forma mais ou menos abstracta, optámos por concretizá-lo com o apoio de uma linguagem de programação. A nossa escolha recaiu sobre o Visual Basic 3.0 da Microsoft que é um produto, pelas suas potencialidades, verdadeiramente excitante. A abordagem que faremos possibilitará aos leitores apreenderem os conceitos de «objecto», «método» e «propriedade» e utilizarem, com conhecimento de causa, os objectos que o Visual Basic coloca à disposição dos programadores, ou seja, escreverem programas que usam objectos. As vertentes da criação dos objectos e seus mecanismos internos ficam reservadas para outro ciclo de artigos, de natureza muito mais técnica, onde a linguagem de suporte

usada será o Borland C++ 3.1.

Sobre o «Visual Basic e Seus Objectos Amestrados» digo-vos já que o assunto dará «pano para mangas» e estará por aqui nos próximos números da **Spooler** (é imprevisível dizer, neste momento, quantos). Por agora vamos ficar com alguns conceitos teóricos, menos interessantes (mas mais importantes) que tudo o que estará para vir.

OS MÉTODOS TRADICIONAIS DE PROGRAMAÇÃO

Um programa de computador serve para efectuar um conjunto de acções sobre um lote de dados, de forma a produzir um determinado resultado. Os programas são tipicamente contruídos em redor das tarefas a desempenhar: pensa-se (e muitas vezes escreve-se) primeiro o que o programa vai fazer, relegando para segundo plano as estruturas de dados.

A abordagem tradicional do desenvolvimento de um programa segue a metodologia *top-down*: parte-se de um programa que se pretende que desempenhe uma tarefa de larga envergadura e vai-se fraccionando esta sucessivamente noutras tarefas mais simples; cada tarefa assim determinada ainda pode ser fraccionada, e prossegue-se assim até se obterem, idealmente, os conjuntos de instruções básicas que irão corresponder a instruções codificáveis na linguagem de implementação escolhida (Pascal, C, Cobol, Fortran, etc).

As desvantagens desta abordagem têm a ver com a dificuldade de reutilizar o código escrito: como as soluções implementadas têm em vista um determinado objectivo, os procedimentos e funções escritos são, quase sempre, de «vistas curtas» e, muito possivelmente, terão de ser alterados para satisfazer os requisitos de um novo programa.

OS NOVOS MÉTODOS DE PROGRAMAÇÃO

Contrariando a técnica de escrever um programa onde todas as rotinas se baseiam na resolução de um problema, a «Programação Orientada por Objectos» desenvolve o programa em torno dos dados: até uma fase bastante avançada o código escrito para manipular os dados não está comprometido com o objectivo final do programa. A generalidade assim adquirida torna o código muito estanque e autónomo, transmitindo-lhe grande probabilidade de poder ser reutilizado sem alterações. Assim, e por oposição à escrita de um programa que tem uma finalidade e, por isso, tem de manipular informação de determinada forma, constrói-se um sistema onde a informação oferece diversas facilidades e capacidade de manipulação, em redor das quais se constrói um programa para atingir um determinado objectivo.

O QUE É A PROGRAMAÇÃO ORIENTADA POR OBJECTOS?

A programação orientada por objectos (também designada vulgarmente por OOP=Object-Oriented Programming) é uma metodologia de desenvolvimento de programas para computador. O seu objectivo é permitir aos programadores criar melhores programas em menos tempo, potenciando a reusabilidade do código, ou seja, possibilitando que parcelas de programas escritos no passado sejam incorporados em novos programas. Para alcançar esta meta recorre-se à estratégia de conjugar numa única entidade os dados e as partes de programa necessárias para manipular esses dados. A esta entidade, que incorpora simultaneamente o código e os dados, chama-se objecto.

Os objectos têm a particularidade de serem «criaturas misteriosas», a que o programador sabe recorrer (sabe como comunicar com o objecto e conhece as suas funcionalidades) mas sobre o qual não conhece as complexidades de funcionamento – não conhece os mecanismos interiores. Se isto pode ser, aparentemente, uma desvantagem («se não sei como funciona a máquina não lhe posso introduzir alterações») é, na realidade, uma enorme vantagem: uma vez liberto das preocupações de implementação e funcionamento dos objectos, o programador concentra-se apenas na tarefa de os combinar entre si, colocando-os a funcionar de forma a que comuniquem e trabalhem para si.

De facto, a «Programação Orientada por Objectos» valoriza a relação entre os objectos em detrimento das particularidades de implementação dos ditos cujos.

A abstracção que decorre deste facto é interessante: um objecto passa a ser caracterizado por «aquilo que pode fazer» em vez de o ser «pelo modo como funciona». Ao programador basta conhecer o modo de interagir com o objecto para que a sua satisfação seja completa.

O termo «método» surge vulgarmente associado aos conceitos de Programação Orientada por Objectos. Os métodos pertencem aos objectos e são o que foi já referido como «o código associado aos dados». Se nos afastarmos da área da informática e tentarmos imaginar um objecto como algo com substância e inteligente, poderemos dizer que os dados correspondem aos conhecimentos que o objecto alberga dentro de si e os métodos constituem o que o objecto pode e sabe fazer, incluindo o acesso aos conhecimentos. Uma das características mais curiosas dos objectos é o facto de serem muito «ciosos» dos seus dados: o programador não pode aceder directamente aos dados; pode, quanto muito, evocar os métodos que o objecto oferece para manipular esses dados!

«De facto, a «Programação Orientada por Objectos» valoriza a relação entre os objectos em detrimento das particularidades de implementação dos ditos cujos.»

O TRABALHO COM OS OBJECTOS

Poderemos considerar que existem dois níveis de trabalho com objectos:

a) um nível baixo, extremamente técnico, onde se constroem os objectos, definindo as estruturas de dados e escrevendo o código para manipular os dados; o grau de abstracção para trabalhar a este nível tem de ser muito grande: pretende-se que o trabalho produzido seja o mais genérico possível; para trabalhar a este nível recorre-se normalmente ao C++.

b) um nível alto, muito pouco técnico, onde o programador desempenha praticamente o papel de operário especializado, conhecedor da funcionalidade dos objectos, sabendo como pô-los a funcionar e a trabalhar para si; o desenvolvimento de aplicações a este nível é muito rápido e os resultados são normalmente espectaculares; para trabalhar a este (alto) nível recorre-se a modernas linguagens de desenvolvimento, como o Visual Basic.

OS OBJECTOS DENTRO DO VISUAL BASIC

O VISUAL BASIC constituiu o grande contributo da Microsoft no campo da «Programação Orientada por Objectos». Como já se constatou, um objecto é algo de muito abstracto, normalmente uma estrutura de dados genérica com um determinado conjunto de capacidades associado. Pois dentro do VISUAL BASIC os objectos têm substância (e como têm!): os objectos são aquilo que se vê no ecrã (os botões, as caixas de texto, as figuras), sendo um programa «escrito» muito simplesmente à custa da disposição de um conjunto de objectos numa folha de papel (já agora, a esta folha de papel chama-se *form* – aparecerá aos nossos olhos como uma janela, dentro da qual está o programa a funcionar – e é também um objecto). Estes objectos vão ter diversas propriedades configuráveis – a cor, a dimensão, a posição no ecrã – e vão ter a capacidade de reagir à interacção do utilizador final do programa.

O VISUAL BASIC constituiu o grande contributo da Microsoft no campo da «Programação Orientada por Objectos».

O papel do programador de VISUAL BASIC é conhecer os objectos à sua disposição, saber como os utilizar e o que pode fazer com eles, e escrever os pequenos troços de programa que forem específicos da aplicação que está a desenvolver. O trabalho com o VISUAL BASIC é desenvolvido essencialmente na área do desenho (desenhando-se a aparência do que é mostrado no ecrã) em vez da área da codificação.

AS HERANÇAS (INHERITANCE)

A herança é a capacidade-chave para que os objectos sejam reutilizáveis de programa para programa. Quando um objecto A «herda» as capacidades de outro objecto B, todas as propriedades e métodos de B estarão imediatamente disponíveis ao objecto A. Concretizemos: suponhamos a existência de dois objectos A (um *form*) e B (um botão trivial, desses que aparecem em todas as aplicações de Windows). O objecto B tem as propriedades ETIQUETA (o texto que aparece escrito no botão) e DISPONÍVEL (a propriedade que configura o botão para poder ser utilizado ou estar indisponível). Se o objecto B (botão) for colocado no objecto A (*form*), o objecto A vai «herdar» todas as capacidades do objecto B. Isto significa que o objecto A passa a «saber» como alterar o texto escrito no botão e como colocá-lo disponível ou indisponível. Por exemplo:

```
A!B.ETIQUETA    =  "Imprimir"   'escreve o texto «Imprimir» no botão
A!B.DISPONÍVEL  =  FALSE        'coloca o botão indisponível
```

O que sucedeu é que um objecto generalista, como o botão, foi incorporado num *form* para obter um determinado resultado concreto pretendido pelo programador. O *form* vai herdar do mesmo modo todas as capacidades de todos os objectos que forem nele colocados.

PROGRAMAÇÃO CONTROLADA POR EVENTOS

A elaboração de um programa em VISUAL BASIC insere-se numa filosofia nova e particular das técnicas de programação, que é a «programação controlada por acontecimentos». Contrariamente às técnicas tradicionais de escrita de um programa, onde existia o conceito de programa principal funcionando em ciclo e dentro do qual o utilizador interagia com o programa para obter os resultados pretendidos, um programa feito em VISUAL BASIC é constituído por uma série de fragmentos de programa, cada um escrito para responder a uma situação específica: um «clique» sobre um botão, o «arrastamento» (*drag*) de uma janela, a pressão de uma determinada combinação de teclas, etc.

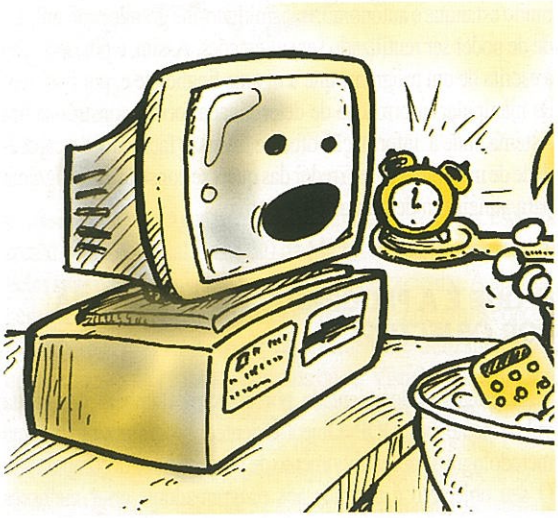
Na realidade, os objectos de um programa de VISUAL BASIC comunicam entre si, com o Windows e com o utilizador através de «eventos». Quando nada acontece num programa de VISUAL BASIC (o utilizador não age sobre a janela do programa e não está a ser processada informação) a aplicação encontra-se num estado de letargia, designado normalmente por *idle* – o programa está rigorosamente PARADO. Alguns eventos não são geridos pelo programador, dizendo apenas respeito ao sistema operativo (um exemplo disto tem a ver com a manipulação das janelas, que é totalmente gerida pelo Windows).

Como referimos inicialmente, estas linhas que acabaram de ler abrem um ciclo de artigos que publicaremos nos próximos números da **Spooler**, dedicados à programação orientada por objectos com o Visual Basic 3.0. Neste sentido, abordaremos o modo de usar o ambiente de trabalho do Visual Basic, as estratégias de concepção de programas nesta linguagem, a sintaxe e semântica da linguagem BASIC e a funcionalidade dos objectos oferecidos ao programador.

Acompanhando esta série de artigos, onde a profundidade dos temas irá aumentar progressivamente, apresentaremos sempre um conjunto de programas exemplo, com o código fonte, onde estarão aplicados, na prática, os temas tratados no artigo. Como bibliografia aconselhada (e também consultada pelo autor destas linhas) sugerimo-vos os excelentes manuais da Microsoft que acompanham o produto (são, sem favor nenhum, verdadeiramente magníficos) e/ou o livro «Do-it-yourself VISUAL BASIC for Windows» da editora SAMS Publishing (passe a publicidade).

Já chega por agora – esperemos que este assunto seja OBJECTO da vossa melhor atenção. Até ao próximo número.

Paulo Bernardo



FONTES

colaboração do leitor

Nunca é de mais insistir-se em temas que consideramos importantes. Esta matéria, se bem que não da mesma forma, já foi abordada o mês passado. Mas António Mendoça acrescenta-lhe a utilização do «Sopa de Letras» de Armando Frazão, que tanto sucesso tem tido entre a comunidade Spooler.

Esta pequena UNIT feita em Turbo Pascal, utiliza os ficheiros criados pelo programa «Sopa de Letras» publicado na **Spooler Nº 22**, sendo assim possível utilizá-las nos seus programas dando-lhes um aspecto diferente como o que se vê em alguns programas bem conhecidos de toda a gente, tais como o CPAV Antivírus, o Norton e o PCTools 8.0.

A UNIT

Esta *unit* tem três procedimentos. O primeiro, talvez o mais fácil de utilizar, tem um único parâmetro: o nome do ficheiro da fonte tal como é exemplificado a seguir:

Procedure LoadFontFile(Nome:string);
Exemplo: LoadFontFile('TESTE.FON');

O segundo procedimento é um pouco mais complexo de utilizar, embora seja mais simples estruturalmente. A sua vantagem é não ser preciso «andar» com os ficheiros das fontes atrás do programa visto ficarem contidas dentro do executável. Passo agora a explicar a sua utilização e sintaxe.

UTILIZAÇÃO

Em primeiro lugar há que fazer uma pequena modificação no ficheiro da fonte, a qual consiste em retirar o primeiro byte do ficheiro que contém o número de bytes que cada carácter ocupa. Para isso criei o programa CONVFON que tem a seguinte sintaxe:

CONVFON ficheiro.fon ficheiro.bin

Este programa diz-lhe também o número de bytes que ocupa cada carácter, visto que vai necessitar desse valor mais tarde, como irá perceber. Depois da fonte modificada há que convertê-la para um ficheiro objecto (.OBJ) utilizando o utilitário BINOBJ que, para quem não sabe utilizar, tem os seguintes parâmetros:

BINOBJ <ficheiro[.BIN]> <ficheiro[.OBJ]> <public name>
PUBLIC NAME – é o nome a dar ao procedimento para depois se proceder ao *link*.
Atenção: este nome não pode conter espaços.

Exemplo: BINOBJ TESTE.BIN TESTE.OBJ TESTE_FON

Sintaxe:

Procedure LoadLinkFont(varBuffer:pointer; NumBytes :byte);

O BUFFER neste caso vai ser o procedimento que criámos com o BINOBJ, aliás este método é idêntico ao de *link* das fontes gráficas e *drivers* do Turbo Pascal. Se surgirem dúvidas aconselho a dar uma vista de olhos pelos exemplos que vêm normalmente com o TP. O parâmetro NumBytes é o número de bytes que ocupa cada carácter obtido pelo método explicado a cima.

Depois deste trabalho todo podemos utilizar a fonte usando o seguinte método:

```
Program Teste;
...
...
{$L TESTE.OBJ}
Procedure TESTE_FON;External;
...
...
Begin
...
...
{ e aqui chegamos ao fulcro da questão }

LoadLinkFont(@TESTE_FON,16);
...
...
end.
```

O último procedimento que, no fundo, são três, destinam-se a por tudo de volta ao normal e não têm quaisquer parâmetros. Os seus nomes são:

```
Procedure LoadRomFont8x16;
Procedure LoadRomFont8x14;
Procedure LoadRomFont8x8;
```

E assim acabo esta pequena mas útil utilização dos ficheiros criados pelo programa «Sopa de Letras» feito pelo Armando Frazão. Espero que façam bom uso destes procedimentos.

Esta pequena UNIT feita em Turbo Pascal, utiliza os ficheiros criados pelo programa «Sopa de Letras» publicado na *Spooler* Nº 22

António Mendoça

colaboração do leitor

SUPER TESTE

Não vão faltar, certamente, leitores interessados neste curioso programa do nosso Miguel Gonçalves. Curiosos (se surpreendentes...) também os resultados obtidos com o exemplo que o autor incluiu. Esperamos que se divirtam e não fiquem complexados...

> **"Foi ao recordar-me do tempo dos Spectrum 48K que tive a ideia para este programa."**

Foi ao recordar-me do tempo dos Spectrum 48K que tive a ideia para este programa. Nessa altura, para treinar os meus recém adquiridos conhecimentos de programação, eu pegava em testes aos mais variados aspectos da personalidade de um indivíduo que sempre iam aparecendo em jornais e revistas e informatizava-os, de forma a saber imediatamente os resultados, sem estar a contar pontos ou a consultar tabelas classificativas.

Ocorreu-me a possibilidade de construir um pacote de 10 ou 20 testes que seriam executáveis, mas achei que esse tipo de programas só teria interesse enquanto nem todos os testes fossem feitos, portanto, a sua «duração» seria bastante reduzida pois todos sabemos qual é o destino dos programas que não interessam!

Decidi então fazer o Super Teste, cuja função é construir novos testes ou questionários mediante o gosto pessoal dos seus criadores que, pelo modo como este programa foi concebido, ficam com um amplo espaço de manobra para mostrarem as suas capacidades.

INSTALAÇÃO

Para instalar o Super Teste basta seguir os seguintes passos:

- **Criar um directório para o programa num disco ou disqueta;**
- **Copiar os ficheiros referentes ao programa da disqueta da Spooler para o directório acabado de criar;**
- **Verificar se tem o *device* ANSI.SYS instalado no ficheiro CONFIG.SYS da raiz da sua drive de arranque;**
- **Se não estiver, use um editor (como, por exemplo, o EDIT do MS-DOS 5) para escrever numa última linha do CONFIG.SYS:**

DEVICE=C:\???\ANSI.SYS
(onde está ??? deve estar o directório onde tem instalado o DOS)

CRIANDO O SEU PRIMEIRO TESTE

Para dar o máximo de liberdade aos criadores de testes, decidi que os ecrãs com as perguntas, introduções, resultados, etc. fossem criados em programas apropriados para esse efeito, nomeadamente, editores ASCII (como o EDIT ou QEDIT) ou editores ANSI (como o ANSPAINT ou o THEDRAW) com as suas capacidades de cor, animação e música.

Para fazer um teste deverá criar ficheiros/ecrã com um máximo de 80 colunas por 25 linhas (caso contrário passarão os limites do ecrã) com cada pergunta, ecrã de introdução, ecrã final (opção) ou resultado. Após a criação de cada um destes ficheiros (em ANSI e/ou ASCII, como expliquei) poderá então correr o Super Teste que, mediante as informações a serem dadas, montará um executável.

Ao entrar no programa escrevendo STEST seguido do caminho (*path*) completo do directório onde se encontram os ficheiros/ecrã (caso seja o directório actual não é preciso indicar nada) (ex: STEST C:\STEST\ANSFILES), após a apresentação, ser-lhe-ão imediatamente perguntadas informações acerca do teste que pretende criar (como o número de perguntas, número de hipóteses de resposta que o utilizador terá por pergunta, cores para os ecrãs ASCII, etc). As várias informações que o programa requer estão divididas em 5 grupos diferentes:

- Informações Gerais: Perguntas sobre as características do teste a ser criado;
- Pontuações: Valor em pontos de cada resposta do utilizador;
- Introdução e Fim: Nomes dos ficheiros/ecrã de introdução e final (facultativo);
- Classes e Resultados: Atribuição dos diferentes ficheiros/ecrã de resultado por pontos que o utilizador consiga realizar;
- Perguntas: Relação entre o número da pergunta e o ficheiro correspondente.

O Super Teste obterá todas estas informações por meio de dois tipos de campos diferentes que são os numéricos e os nomes de ficheiro. Ao preencher um campo numérico terá disponíveis as teclas do cursor para incrementar ou decrementar o número, a tecla ESC para sair do programa (caso confirme), as teclas F1 e F2 para saltar para o princípio do programa ou para o princípio da secção actual (respectivamente) e a tecla ENTER para finalizar. Por seu lado, ao preencher um campo com o nome de um ficheiro poderá usar todas as letras para escrever o nome, as teclas do cursor ESQUERDA e DIREITA, BACKSPACE para apagar, ESC para sair (caso confirme), a tecla do cursor CIMA para voltar ao início da secção e a tecla do cursor BAIXO ou ENTER para finalizar. Ao chegar ao fim ser-lhe-á mostrada a evolução da criação do executável após a qual sairá para o sistema operativo.

Para que veja tudo isto a funcionar está disponível um teste comprimido no ficheiro autodescompactável EXEMPLO.EXE que contém ficheiros/ecrã para a montagem de um teste. Siga as instruções dadas no ficheiro LEIA-ME para saber como deve proceder. Caso queira compilar as fontes deste programa consulte primeiro o ficheiro COMPILAR.TXT para mais informações.

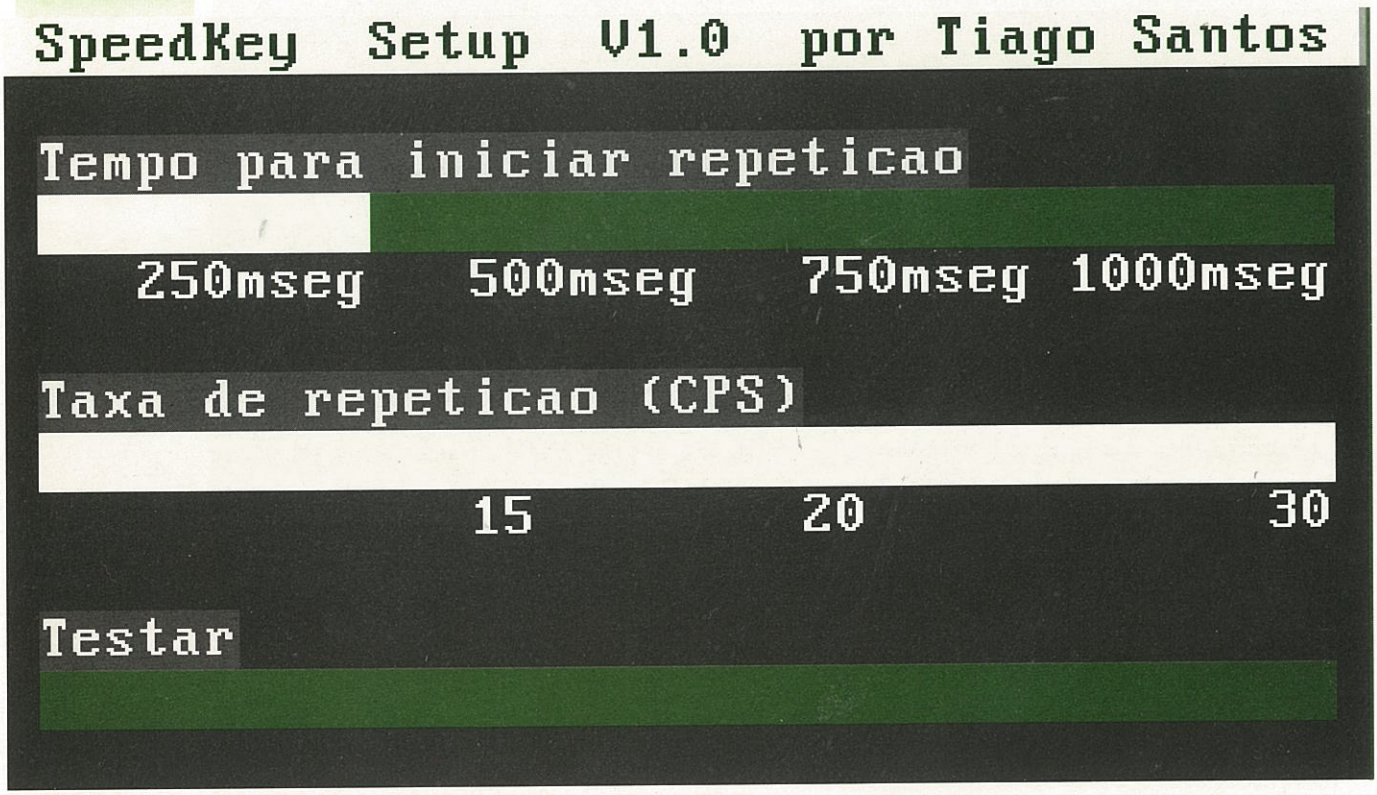
Esperando que o Super Teste seja do seu agrado...

Miguel Gonçalves

Graficria
PUBLICIDADE E ARTES GRÁFICAS, LDA.
QUALIDADE
EFICIÊNCIA
RAPIDEZ
Rua Duarte Pacheco Pereira, 69-A – Damaia-de-Baixo – 2700 AMADORA – Telefs.: 474 60 11/41 – Fax: 475 22 89

colaboração do leitor

SPEEDKEY



Quem usa com frequência processadores de texto ou similares sabe bem a necessidade de deslocar o cursor no ecrã com uma certa rapidez para se alcançardeterminada posição. Foi por este motivo que criei o SPEEDKEY, o qual contém três ficheiros, que passo a descrever:

SKSETUP.EXE – Este programa quando executado mostra no ecrã um quadro com duas barras vermelhas. Referindo-se a primeira ao tempo que o MS-DOS leva a iniciar a repetição e que pode variar entre 250 mseg e 1000 mseg. Esta primeira barra movimenta-se usando as teclas HOME e END. A segunda barra refere-se à velocidade com que um carácter é repetido no ecrã e que pode variar entre 15 e 30 caracteres por segundo. Para movimentar esta barra usam-se as teclas PgUp e PgDn.

Portando, podemos concluir que o modo mais rápido é a barra que indica o tempo de repetição estar em 250 mseg e a barra que se refere à taxa de repetição (caracteres por segundo), estar posicionada em 30. À medida que vamos configurando a velocidade do nosso teclado podemos ir também testando; para tanto basta pressionar continuamente uma tecla. Para sair do SKSETUP basta primir ESC, e este automaticamente criará um ficheiro chamado SPEEDKEY.DAT.

SPEEDKEY.EXE – Este programa serve para ler o SPEEDKEY.DAT, anteriormente criado pelo SKSETUP.

SPEEDKEY.DAT – Este é o ficheiro no qual são guardados os dados do SKSETUP para serem lidos posteriormente pole SPEEDKEY.

O SPEEDKEY foi escrito em Pascal e, para ajustar a taxa de repetição das teclas e o tempo para iniciar a repetição, usei a Função 3 da Interrupção 16.

Espero que o SPEEDKEY vos seja útil tal como tem sido a mim.

Tiago Santos

O FANTÁSTICO MUNDO DAS BBS

PARTE VI

Como prometido no nosso artigo anterior, vamos hoje enfrentar um dos argumentos mais «escaldantes» do mundo das BBS: a transferência de ficheiros.

A magia desse processo não deixa de nos surpreender, mesmo depois de tanto tempo. Pensemos um momento no seu significado: eu tenho um programa no meu computador e quero que o meu amigo José partilhe comigo o interesse pelo mesmo. Assim, através da transferência de ficheiros eu vou poder transmitir, enviar, copiar (inventem o termo que preferem) para o computador dele esse tal ficheiro.

Não se trata de uma cópia em disqueta, não se trata de uma cópia através de uma rede (ou *network*) mas por via telefónica, esse é o grande desafio das BBS, como dissemos logo no primeiro artigo: trabalhar através das linhas telefónicas.

Não há transferência de matéria, trata-se de um fluxo de dados que percorre essa tal linha partindo do meu disco, é lido na memória, por meio de um protocolo pre-establecido é enviado para o meu modem, o meu modem codifica os dados e envia-os como sinais reconhecíveis pelo sistema telefónico ao modem de destino, esse descodifica os sinais e, através do mesmo protocolo de partida, escreve os dados na memória do computador e logo no disco.

Resultado: em poucos minutos uma cópia exacta do programa que estava no meu PC encontra-se no PC do José. Fantástico!

Vamos então ver a operação de transferência de ficheiros através dos seus comandos mais usuais.

Em termos telemáticos as transferências de ficheiros dividem-se em duas grandes categorias os «uploads» e os «downloads».

Acho muita graça a essa definições porque faz-me lembrar a diferença que existe entre Subida e Descida: nenhuma, ou seja tudo é relativo, depende de quem estiver a olhar. No processo de transferência de ficheiros a definição tem o mesmo significado. Do ponto de vista do Sysop um *download* é uma transferência da **sua** BBS para o sistema remoto (do *user*) e um *upload* é uma transferência (recepção) de um ficheiro do sistema remoto (do *user*) **para** a BBS. Em termos técnicos as duas operações são estritamente idênticas, o resultado é que é oposto.

Vamos portanto analisar o processo, neste artigo, pondo-nos na óptica do **utilizador** que é o leitor do artigo, assim:

Download: Transferência de um ficheiro da BBS para o meu sistema (puxar ficheiros – em linguagem normal)

Upload: Transferência de um *ficheiro* do meu computador para a BBS (envio de ficheiro – em linguagem normal)

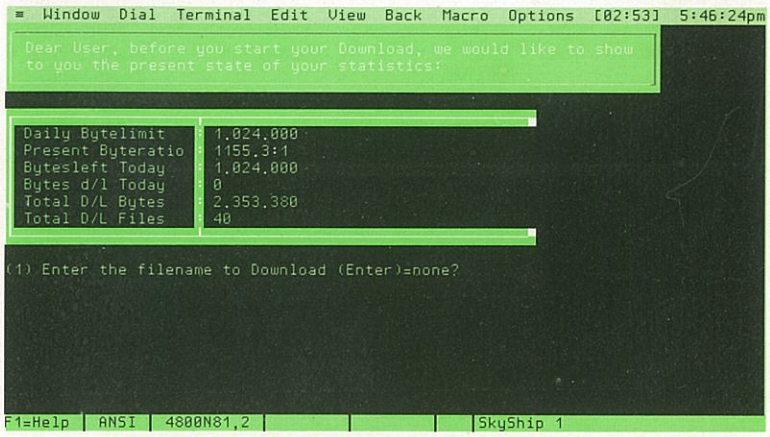
COMO FAZER *DOWNLOADS*?

Para fazer o *download* de um ficheiro terá, o utilizador, que dar o nome do ficheiro (ou dos ficheiros) que deseja receber. Para ajudar na procura do nome exacto do ficheiro que vai ser transferido, poderá utilizar os comandos de busca que foram analisados no artigo precedente:

L: Locate ficheiros (por nome, por exemplo L SCAN*.*)
N: New ficheiros (ou seja mostra-me lá todos os ficheiros novos a partir de uma certa data e nos directórios que eu escolher a seguir).
Z: Procura por texto incluído no título ou descrição (por exemplo Z VIRUS A: procura todos os ficheiros que incluam o texto VIRUS no título ou na descrição).
F: Mostra todos os directórios disponíveis na BBS dos quais é possível «puxar» os ficheiros.

Para concretamente fazer o download terá que dar o comando **D** (para um único ficheiro) e o comando **DB** (*download batch* – para vários ficheiros). Depois de dar esse comando, a BBS responde qualquer coisa parecida com este *prompt*:

1) Enter the file name to Download (Enter) =none?



> **Como prometido no nosso artigo anterior, vamos hoje enfrentar um dos argumentos mais «escaldantes» do mundo das BBS: a transferência de ficheiros.**

Neste ponto terá que introduzir o nome do ficheiro, poderá também escrever vários ficheiros um atrás do outro. Se o ficheiro for encontrado, aparecerá uma informação sobre o número de bytes de que consta o ficheiro e uma estimativa do tempo necessário para o processo de *download*. Quando terminar de introduzir os nomes dos ficheiros carregue no Enter. Depois de introduzir os nomes todos (ou o nome se for um único ficheiro), está pronto para começar a transferência.

```
= Window Dial Terminal Edit View Back Macro Options [03:16] 5:46:48pm

 Daily Bytelimit    : 1,024,000
 Present Byteratio  : 1155.3:1
 Bytesleft Today    : 1,024,000
 Bytes d/l Today    : 0
 Total D/L Bytes    : 2,353,380
 Total D/L Files    : 40

(1) Enter the filename to Download (Enter)=none? newfiles
Checking file transfer request.  Please wait, Carlos ...
(1)   NEWFILES.ZIP    9073 bytes,  0.7 minutes (approximate) FREE / NO TIME

(2) Enter the filename to Download (Enter)=none?

Batch Download Time:  0.7 minutes (approximate)
Batch Download Size:  9073 bytes (9 blocks)
Batch Protocol Type:  Ymodem/G
(Ready to Send in Batch Mode)

(A)bort, (E)dit, (G)oodbye, (L)ist, (P)rotocol, (Enter)=continue? ( )
F1=Help | ANSI | 4800N81,2 |          |          |  |SkyShip 1
```

A maioria dos programas de comunicações vem preparados com uma série de protocolos de transferência e, portanto, basta que o protocolo escolhido na BBS seja o mesmo seleccionado no seu terminal. Lembre-se que:

1. Se não tiver o mesmo protocolo seleccionado, a transferência nunca vai começar.

2. Se o seu teminal não estiver configurado para o protocolo que escolheu, a transferência nunca vai começar.

3. A velocidade de transferência depende de inúmeros factores como: velocidade do modem, estado da linha telefónica, tamanho do ficheiro, escolha de protocolo.

A propósito deste último factor, ou seja do protocolo, prefiro dedicar um artigo à análise dos vários protocolos existentes, ou pelo menos dos mais comuns.

Nesta instância apenas assinalo os que são de maior uso e consumo:

ZMODEM
Y-MODEM
Y-MODEM/G
HSLINK
BIMODEM
XMODEM

```
= Window Dial Terminal Edit View Back Macro Options [02:16] 5:45:48pm

[ ] Main Menu [ ] Mail Menu [ ] Special Options [ ] User Utilities

FILES Menu Command ('MENU' for options)? t

   (A) Ascii (Non-Binary)
   (X) Xmodem (Checksum)
   (C) Xmodem CRC
   (F) 1k-Xmodem/G
   (G) Ymodem/G
   (H) HS/Link (Bi-Directional Batch)
   (O) 1k-Xmodem
   (Y) Ymodem
   (Z) Zmodem (Batch U/1 D/1)
   (M) Zmodem w/ MobyTurbo
   (Q) Graphic Zmodem
   (N) None

Default Protocol Desired (Enter)=no change? (G)
F1=Help | ANSI | 4800N81,2 |          |          |  |SkyShip 1
```

Se o seu programa de comunicações não estiver preparado para arrancar automaticamente terá que fazê-lo manualmente, isto é, terá que por exemplo carregar na tecla PgDn ou então nas teclas Alt-R (*receive*) ou ainda nas teclas Alt-D (*download*). Vai tudo dar ao mesmo mas as técnicas dependem intrinsecamente do programa.

Por variadíssimos motivos um *download* pode abortar (outra maneira de dizer «ficar incompleto»). Como veremos noutra altura, os protocolos são suficientemente «inteligentes» para saber onde o processo parou e numa nova ligação retomar exactamente do ponto onde se encontraram antes do *crash*. Ma isso, como disse, é tema para outro artigo.

Outra maneira de fazer *dowloads* «indirectamente» é de percorrer no seu monitor a lista de ficheiros disponíveis e quando aparece um ficheiro que interessa «flagar» o ficheiro. O que é um *flag*? Um *flag* (em português: bandeira) é uma marca que se deixa na lista para que o sistema saiba e perceba que esse ficheiro, mais tarde, depois de termos acabado de ver a listagem, está marcado para *download*. Assim, percorrendo a nossa lista de ficheiros com o comando **F** poderemos «flagar» vários ficheiros que, caso nos interesse, poderemos puxar em seguida.

Há várias coisas que é necessário dizer a propósito dos *downloads*:

1. O número de ficheiros que podem ser puxados (por dia, ou por sessão) depende da configuração que o sysop deu ao nível de segurança a que o utilizador pertence.

2. Independente do ponto 1, a quantidade total de bytes (de um ou mais ficheiros) que podem ser puxadas também depende do nível de segurança do utilizador.

3. Em regra geral, esses dois factores são tanto maiores quanto for maior o nível de segurança, chegando a ser «ilimitado» para certos níveis privilegiados (sysops) ou nulo ou quase nulo para utilizadores muito «verdes».

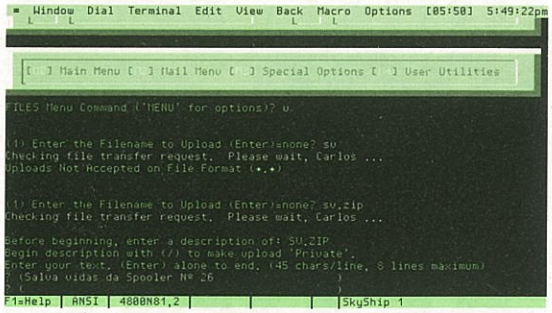
Como tudo na vida, a melhor maneira de se familiarizar com esse processo é o de experimentar e, se falhar... recomece até conseguir; e se ainda não conseguir, deixe um Comment para o Sysop: ele está aí para ajudá-lo a resolver esse tipo de problemas.

COMO FAZER *UPLOADS* (E PORQUÊ) ?

Vamos responder a essa pergunta começando pelo fim: porquê?

Uma BBS vive de muitas coisas: participação activa dos seus utilizadores nas áreas de mensagens, ficheiros, boletins informativos etc. Se coubesse ao Sysop e unicamente a este encher o disco de ficheiros novos, interessantes e «legais», não ganhava para a conta telefónica. Assim, uma maneira clássica de, por um lado contribuir concretamente para a melhoria da BBS, por outro lado (geralmente) para subir de nível de accesso, é o de enviar (*upload*) ficheiros para a BBS.

O processo é idêntico ao do *download* em termos técnicos, mas um pouco diferente em termos práticos.



O ficheiro tem que existir no computador do *user* e não deve existir na BBS. Escolhe a opção **U** e o sistema (em regra geral) despeja para o ecrã uma mensagem onde diz quais as condições em que o ficheiro será aceite pelo sistema (vamos ver isso já a seguir) logo pergunta o nome do ficheiro e, se não existir um ficheiro com o mesmo nome já na BBS, então pergunta uma breve descrição do mesmo para que conste na lista e para que os outros utilizadores possam perceber do que se trata.

O *user* que vai fazer o *upload* deve tomar várias precauções, sendo as mais básicas as seguintes:

1. Nunca enviar ficheiros comerciais;
2. Nunca enviar ficheiros cujo conteúdo é ignorado pelo utilizador;
3. Verificar na medida do possível que o ficheiro não contenha vírus e/ou troianos;
4. Ter em atenção as recomendações feitas pelo sistema (pelo sysop) antes da transferência.

A esse propósito e em regra geral não devem ser enviados para uma BBS ficheiros comerciais, pornográficos, ficheiros com *copyright* que proíbe a sua divulgação, ficheiros antigos que já não interessam a ninguém. Podem (e devem) ser enviados somente ficheiros ***shareware, freeware, public domain*, demonstrações e imagens não pornográficas.** Lembre-se a propósito desse último ponto que o seu filho menor poderia ter accesso a material que não é conveniente e por isso, se existir um directório de ficheiros desse género ele deverá ser sempre em forma privada, porque na privacidade tudo é permitido mas publicamente as coisas são diferentes.

Bom, depois de fazer o *upload*, o sistema vai analisar o seu ficheiro e decidir se o vai aceitar ou não, dependendo de todos os factores acima mencionados. Assim e se o ficheiro passar o teste, terá o prazer de ter contribuído para o crescimento da sua BBS preferida.

Até para o mês que vem!

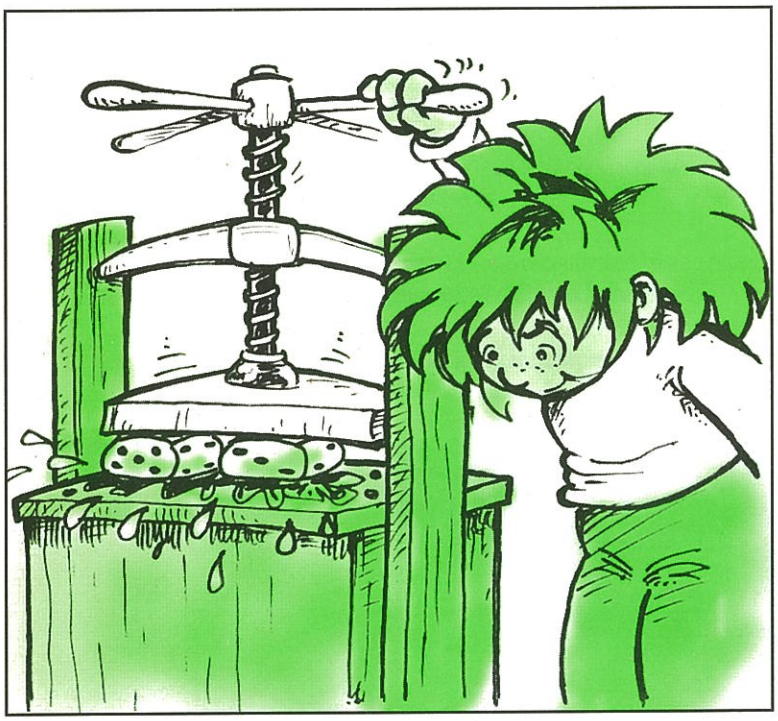
Mário Pozzetti, o vosso amigo Sysop

```
≡ Window Dial Terminal Edit View Back Macro Options [07:22] 5:50:54pm
Thanks for the file(s), Carlos!

This system is set to test uploaded files.  This can be a lengthy process.

(H)ang up now, or press any other key to remain online..........

Verifying SV.ZIP...
    UpLoadProcessor 1.04β          Registered to: SKYSHIP BBS
Copyright 1992-93  Stacy Smith     Serial number: 1129

   Identifying archive format...ZIP
   Screening archive for corruption...OK
   Unpacking archive...OK
   Testing files for viruses, etc...OK
   Checking for duplicate files...OK
   Checking age of files...OK
   Packing archive...
   Commenting archive...OK
   Archive passed.

Press (Enter) to continue?
F1=Help | ANSI | 4800N81,2 |          |          |  |SkyShip 1
```

colaboração do leitor

MÉTODOS DE COMPRESSÃO DE DADOS

PARTE I



Saudações a todos os programadores (amadores e profissionais) e gentes ligadas à Informática. Tenho 14 anos e programo em Turbo Pascal há cerca de dois anos, podendo-se dizer que domino (bastante) bem a «fera». Também me estou a iniciar na programação em C, que estou a achar bastante interessante.

Fiz a minha «estreia» como colaborador da **Spooler**, a qual pelo seu papel fundamental (cada vez mais) na divulgação e intercâmbio de conhecimentos de Informática, se distingue e destaca entre tantas outras como a melhor de sempre feita em Portugal – as minhas felicitações. Hum, aaahh, como estava a comentar (antes de me deixar levar pela emoção!), fiz a minha estreia como colaborador da **Spooler** neste número, apresentando o primeiro artigo de um total de 4, cujo tema é, tal como o título indica, métodos (ou técnicas) de compressão de dados.

Cada artigo será acompanhado por um programa demonstrativo para cada método e pelo seu código-fonte em Turbo C na disqueta do respectivo número, exemplificando o método descrito no artigo. Cinco métodos serão examinados :

1º Artigo – Breve Introdução à compressão de dados;
O método de «Compressão de Fileira»;
O método de «Codificação Diatómica»;

2º Artigo – O método de «Mapeamento de Bit»;
O método de «Compressão de Meio Byte»;

3º Artigo – O método de «Codificação de HuffMan»;

4º Artigo – Comparação entre os métodos;
Escolha do que mais se adapta a cada tipo de situação;
Programa Compressor final (compressão auto-adaptável);

Eventualmente será escrito um 5º Artigo sobre o método de compressão «Lempel-Ziv-Welch». Espero que apreciem o meu trabalho.

BREVE INTRODUÇÃO

A razão que leva um utilizador a comprimir os seus dados é bastante óbvia: diminuir o espaço ocupado pelos mesmos, e consequentemente, aumentar o espaço livre. Isso poupa a muitas empresas bastante dinheiro que, de outra forma, teria de ser aplicado inutilmente na compra de suportes para esses dados (nomeadamente discos fixos e\ou flexíveis).

Mas não só... A compressão de dados é também utilizada para reduzir o tempo de transferência de dados (especialmente por via telefónica), e ainda (embora menos usualmente), para fins de segurança, na medida em que a maior parte dos métodos de compressão codificam os dados originais.

A compressão de dados, ao contrário do que muitos possam pensar, não é recente. Já na década de 60 existia, embora os métodos existentes nessa altura possam considerar-se hoje como obsoletos, inadequados e mesmo «rudes». Na realidade, a metodologia de compressão de dados só sofreu um grande desenvolvimento nesta última década.

Uma das primeiras técnicas de compressão de dados a ser utilizada foi a de supressão de caracteres nulos (espaços), que não irá ser discutida aqui (bem como outras técnicas), pela sua simplicidade e pouco (ou nenhum) uso hoje em dia.

Actualmente, *softwares* de compressão de dados como o PKZIP e o LHARC, (*shareware*, por sinal) entre os mais conhecidos, utilizam vários algoritmos de compressão de acordo com as características do ficheiro a comprimir, de forma a optimizar a compressão. Para além destes existem ainda os que comprimem totalmente um disco duro como o Stacker (da Stac Electronics) e, mais recentemente, o DoubleSpace (incluído no MS-DOS Ver 6.0). Todos eles utilizam o método *Lempel-Ziv-Welch* como método principal, por ser bastante eficiente e rápido.

A compressão de dados tornou-se praticamente indispensável nos nossos dias, e qualquer pessoa ligada de alguma forma à Informática já, pelo menos, ouviu falar nela.

Finalmente, vamos ao que interessa!...

O MÉTODO DE COMPRIMENTO DE FILEIRA

A «Codificação de comprimento de fileira» é um método de compressão de dados que reduzirá qualquer tipo de sequência de caracteres repetidos, desde que a mesma alcance um nível pré-definido de ocorrência, ou seja, o número de caracteres repetidos terá que ser maior que o número de caracteres especiais necessários para conter a informação sobre os mesmos, caso contrário, a compressão terá um efeito inverso ao esperado. Aqui vai um exemplo para um melhor entendimento por parte do leitor. Imagine que tem a seguinte linha num ficheiro:

AAAAABBBBCCCCCCDDDEFFFFFFFFFGG

Como poderá reparar, existem bastantes sequências de caracteres repetidos. Para comprimir essas repetições teremos de as substituir por três tipos de caracteres:

IC – carácter Indicador de Compressão;
X – carácter repetido que foi comprimido;
C – Contador de caracteres repetidos
– número de vezes que o carácter se repete.

Note que o carácter indicador de compressão terá de ser um carácter ASCII não utilizado frequentemente (a maior parte dos códigos ASCII acima de 126 não são usados em ficheiros de texto...). Assim a nossa linha seria transformada em:

¤A5¤B4¤C6DDDE¤F9GG

considerando que «¤» é o carácter indicador de compressão. Repare que a nossa linha, inicialmente com 30 caracteres, ficou reduzida a quase metade (18 caracteres).

Não foram comprimidos os caracteres »D»,«E» e«G», pois a compressão dos mesmos, como é óbvio, teria um resultado inverso: aumentaria o número de caracteres existentes, pois os três caracteres necessários para conter a informação são em maior ou igual número aos caracteres a comprimir.

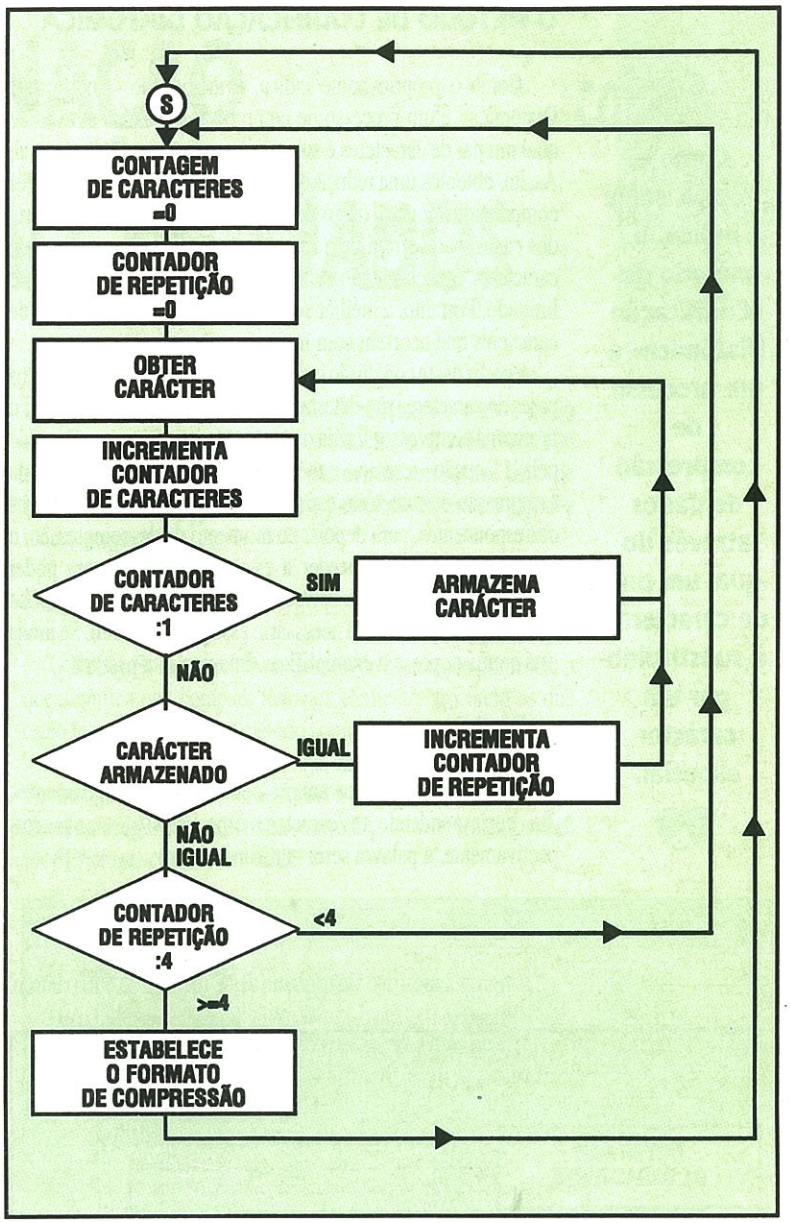


Figura 1

A descompressão, obviamente, utiliza o processo inverso ao da compressão: o carácter especial é identificado (neste caso «¤»), é também identificado o carácter comprimido e, por último, são inseridas as sequências de caracteres repetidos de acordo com o contador de repetição, de modo restaurar o ficheiro original. A figura 1 mostra o fluxograma deste método de compressão.

O programa COMPFIL.EXE (na disqueta deste número) utiliza esse fluxograma para comprimir ficheiros com o algoritmo de compressão de «comprimento de fileira».

Este método é bastante simples, na minha opinião, e ainda utilizado hoje em dia. A sua razão de compressão (tamanho do ficheiro inicial a dividir pelo tamanho do ficheiro depois da compressão) depende totalmente, como deve ter reparado, do número de sequências de caracteres repetidos existentes no ficheiro a comprimir. Passemos agora à análise do método seguinte.

O MÉTODO DE CODIFICAÇÃO DIATÓMICA

Como o próprio nome indica, o método de «Codificação Diatómica» é um processo de compressão de dados através do qual um par de caracteres é substituído por um carácter especial.

Como o próprio nome indica, o método de «Codificação Diatómica» é um processo de compressão de dados através do qual um par de caracteres é substituído por um carácter especial. Assim, obtemos uma redução de 50 % por cada par de caracteres comprimidos. É claro que o ficheiro a comprimir, na maior parte dos casos, não sofrerá uma redução de 50 %, pois o número de caracteres especiais (não usados frequentemente num texto) é limitado. Portanto, a melhor solução será determinar os pares de caracteres que ocorrem com maior frequência.

A partir desta conclusão existem dois caminhos a seguir: ou os pares de caracteres são determinados durante a compressão, ou o programa compressor traz já os pares pré-determinados. Optando pela 1ª opção, teremos que incluir no ficheiro resultante da compressão os caracteres especiais e os seus pares de caracteres correspondentes, para depois, no momento da descompressão, o programa compressor aceder a essas informações para poder descomprimir o ficheiro correctamente, enquanto que optando pela 2ª opção, já não será necessário esse procedimento. Se ainda está confuso, passo a exemplificar. Imagine esta palavra :

Estrela

Considerando que estavam pré-definidos os pares «Es», «re», «la» correspondendo aos caracteres especiais «¤», «¤», «§», respectivamente, a palavra seria transformada em:

¤t§

A figura 2 mostra o fluxograma deste método de compressão.

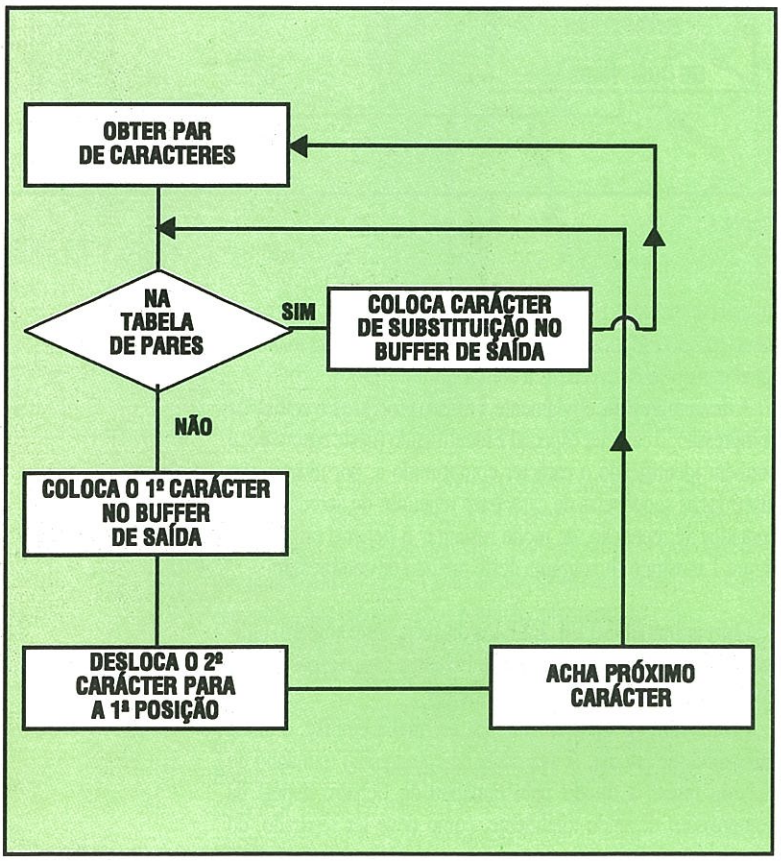


Figura 2

O programa COMPDIAT.EXE (na disqueta deste número) utiliza esse fluxograma para comprimir ficheiros com o método de compressão de codificação diatómica. Este é outro método bastante simples, mas ainda utilizado hoje em dia.

NA DISQUETA...

Na disqueta deste número estão incluídos os programas COMPFIL.EXE e COMPDIAT.EXE ilustrando os dois métodos analisados neste número. Acompanhando os mesmos, os códigos-fonte respectivos em Turbo C e dois ficheiros – EXEMPLO.MCF e EXEMPLO.MCD – para testar cada método. Devido ao facto de os programas terem sido realizados a pensar exclusivamente na demonstração dos métodos aqui apresentados, não existe qualquer verificação de ficheiros, portanto há que ter um cuidado especial nos parâmetros passados na linha de comando, (vulgo *prompt)* para evitar a perda de dados. Os programas de demonstração só comprimem ficheiros de texto; caso tente comprimir qualquer outro tipo de ficheiros, previno-o desde já que é bastante elevada a probabilidade de a descompressão não ser efectuada correctamente. Gostava também de referir que os mesmos não incluem pormenores tais como a duração (em segundos) da compressão e não permitem comprimir mais que um ficheiro de cada vez.

PARA TERMINAR...

Neste primeiro artigo foram analisados dois métodos muito simples e fáceis de entender. Não será tão assim nos números futuros. Para terminar gostaria ainda de salientar que, embora só tenha sido apresentada uma variante do método de «codificação diatómica», existem várias outras. De facto, todos os métodos de compressão têm as suas variantes, que por vezes dão origem a outros métodos. Por razões óbvias essas variantes não serão discutidas aqui.

NO PRÓXIMO NÚMERO

Fica um encontro marcado para daqui a dois meses, onde falaremos de dois métodos de compressão que utilizam a manipulação de bits (se não sabe o que isso é, não se preocupe, pois nessa altura darei uma «lição zinha» sobre o tema). Não esqueça: não perca o próximo episódio de... «Quem encolheu o ficheiro ?!»

José Iria

PARTE I

ORDENAÇÃO E LOCALIZAÇÃO DE DADOS

colaboração do leitor

Desde que os computadores deixaram de ser meras máquinas de calcular super-rápidas para passarem a ter vários usos, o armazenamento de dados tem sido uma das principais tarefas por eles executadas.

Com os computadores mais rápidos e com a utilização destes em diversas aplicações (desde a industrial à doméstica), os métodos de armazenamento de dados têm vindo a evoluir, tentando atingir um máximo de rapidez, eficiência e fiabilidade.

Duas das principais funções relacionadas com o armazenamento de dados são a ordenação e a localização. Neste artigo trataremos a ordenação, ficando a localização para depois.

A ordenação consiste em transformar um conjunto de dados desordenados, numa lista com uma ordem lógica, compreensível por quem a for utilizar. Existem vários métodos para executar a ordenação. Uns mais rápidos, outros mais simples. Analisarei aqui os três mais conhecidos.

Para os exemplos deste artigo decidi utilizar as duas linguagens de programação mais difundidas: a Pascal e a C.

Comecemos então com a ordenação...

ORDENAÇÃO DE DADOS

A ordenação consiste em, como já disse, transformar um conjunto de dados desordenados numa lista com sequência lógica para quem a utilizar. As duas formas de ordenação mais utilizadas são a ascendente e a descendente. Consideremos uma lista ordenada ascendentemente:

E1<=E2<=E3<=...<=En

onde **n** especifica o Nº de elementos da lista. Numa lista ordenada de forma descendente, deparamos com a ordem inversa.

Na ordenação de estruturas de dados singulares as comparações são efectuadas simplesmente entre os elementos a ordenar. Quando trabalhando com os *records* (Pascal) e *structs* (C) utiliza-se apenas uma chave, ou seja, um dos campos, para efectuar a ordenação. É claro que se pode ordenar um conjunto de dados por mais que uma chave, obtendo assim mais listas ordenadas (a ordem dos elementos não será a mesma em todas as listas). Por exemplo, ordenando uma turma de alunos, poder-se-ia utilizar o campo «número» como chave.

Quando as ordenações são aplicadas em listas na memória, efectua-se normalmente uma movimentação de toda a estrutura após a comparação. No caso de ordenações em disco, por motivos de velocidade, é comum movimentar apenas o campo chave para uma tabela de index, associando-o com a posição efectiva da estrutura dentro do ficheiro principal. Mas isto é assunto para o próximo artigo.

Depois duma breve introdução à ordenação, vamos então conhecer o primeiro método de ordenação, o **BubbleSort**. Em todos os exemplos deverá ser considerado um *array* **lista** de inteiros com **count** elementos, *array* esse que deverá ser global a todo o programa.

BUBBLE SORT

O **BubbleSort** baseia-se no método de ordenação por troca, o que significa que compara todos os elementos trocando os que estão fora de ordem. Este método compara todos os elementos da lista com os que lhe seguem, ou seja, de **ndx1** (elemento corrente) a **count** (último elemento), de forma a poder verificar se o poderá trocar com algum deles. A troca ocorrerá quando o elemento **ndx2** for menor que o elemento **ndx1**.

Quadro 1 – Pascal

```
Procedure BubbleSort;

var troca:integer; {serve de intermédio de troca de valores}
    ndx1,ndx2:integer; {elementos de indexação do array}

Begin
    for ndx1:=1 to count-1 do
    for ndx2:= count downto ndx1+1 do
        if lista[ndx1]>lista[ndx2] then
        {compara um elemento com todos os que o seguem na lista}
    begin
        troca:=lista[ndx1];
        lista[ndx1]:=lista[ndx2];
        lista[ndx2]:=troca
        {troca os elementos da lista}
    end;
end; {end BubbleSort}
```

Quadro 2 – C

```
BubbleSort( )

{
    int troca; /* serve de intermédio na troca dos valores */
    int ndx1, ndx2 /* elementos de indexação do array */

    for (ndx1 = 1; ndx1<count; ndx1++)
        for (ndx2=count; ndx2>ndx1; ndx2- -)
            if (lista[ndx1]>lista[ndx2])
            /* compara um elemento com todos os que o  seguem na lista */
            {
            troca = lista[ndx1];
            lista[ndx1] = lista[ndx2];
            lista[ndx2] = troca;
            /* troca os elementos */
            }
}
```

Note-se que, após uma troca, o valor que será utilizado para as restantes comparações será o novo valor e não o antigo (a não compreensão deste aspecto é normalmente o principal factor que leva à dificuldade de assimilar este método). Para o ajudar a entender o **BubbleSort**, apresento de seguida uma simulação de ordenação através deste método:

PASSO Nº	POSIÇÕES NO ARRAY				
	1	2	3	4	5
INICIO	3	7	9	2	5
1	2	7	9	3	5
2	2	3	7	9	5
3	2	3	5	9	7
4	2	3	5	7	9

Tal como pode observar, é um processo demorado (não com dez, mas com muitos elementos). Efectua demasiadas comparações, as quais são a origem da demora. O processo seguinte consegue, utilizando um método diferente, diminuir o número de comparações.

SHELL SORT

Este método, ao contrário do **BubbleSort**, não se baseia na troca de elementos, mas sim num algoritmo que consiste em ir aproximando os elementos da sua posição final, aos poucos e poucos. Para isto, o **ShellSort** utiliza um intervalo (*gap*) entre os elementos que irá comparar. Na última passagem pela lista, o intervalo será 1, de maneira a ordenar os elementos adjacentes.

Quadro 3 – Pascal

```
Procedure ShellSort;

    var gap,ndx1,ndx2,ndx3:integer; {intervalo, posições no array}
        troca:integer; {variável temporária para
        trocas}

Begin
    gap:=count div 2; {gap inicial}
    while (gap>0) do
    begin
        for ndx1:=gap+1 to count do
        begin
            ndx2:=ndx1-gap; {1º elemento a comparar}

        while (ndx2>0) do
            begin
                ndx3:=ndx2+gap; {2º elemento a comparar}
                if lista[ndx2]<=lista[ndx3] then
                    ndx2:=0
                else

                begin
                    troca:=lista[ndx2];
                    lista[ndx2]:=lista[ndx1];
                    lista[ndx1]:=troca;
                    {troca os elementos}
                    ndx2:=ndx2-gap;
                end;
            end;
        end;
        gap:=gap div 2; {obtém novo intervalo}
    end;
end;
```

Assim, se o intervalo for 5, o primeiro elemento será comparado com o sexto, o segundo com sétimo, etc., sendo estes trocados se não estiverem nas posições correctas. Se, por exemplo, a lista estiver a ser ordenada ascendentemente, a comparação do valor 7 com o valor 5 resultará na troca de posições destes dois elementos. Para definir o *gap* inicial, é costume utilizar o valor **n div 2** (Pascal), onde **n** é o número de elementos a serem ordenados. O intervalo irá decrescendo na sua divisão (inteira) por 2, após cada passagem. O código do **ShellSort** pode ser observado nos quadros 4 e 5.

Quadro 4 – C

```
ShellSort( )

{
    int gap,ndx1,ndx2;
    int troca;

    gap=count/2; /* obtém o gap inicial */

    while (gap>0)
    {
        for (ndx1=gap; ndx1<=count; ndx1++)
        {
            ndx2=ndx1-gap; /* 1º elemento a comparar */
            while (ndx2>0)
            {
                ndx3=ndx2+gap; /* 2º elemento a  comparar */
                if (lista[ndx2]<=lista[ndx3]) ndx2=0;
                else
                {
                    troca=lista[ndx2];
                    lista[ndx2]=lista[ndx1];
                    lista[ndx1]=troca;
                    /* troca os elementos */
                    ndx2=ndx2-gap;
                }
            }
    gap=gap/2; /* obtém novo intervalo */
    }
}
```

Numa análise ao código do programa, observa-se que a definição dos elementos a comparar é dada pelas expressões **ndx2=ndx1-gap** e **ndx3=ndx2+gap** onde **gap** é o intervalo entre os elementos. Após obter ambos os elementos, efectua-se a troca de posição dos elementos caso a comparação seja verdadeira. Assim, e após todas as passagens, o *array* deverá ficar ordenado.

Depois de testar estes dois métodos, poderá verificar que a diferença da velocidade de execução é grande, quando operando com muitos números (teste com 1000 elementos).

O método seguinte consegue ser ainda mais rápido, embora a diferença do tempo de execução em relação ao **ShellSort** não seja tão significativa como a do **Shell** em relação ao **Bubble**.

QUICK SORT

O **Quick Sort** é o último método que será analisado neste primeiro artigo dedicado à ordenação e procura de dados. Baseado na ordenação por segmentação, é o mais rápido e eficiente, mas também o mais complexo e difícil de entender (e explicar). Segmentação porquê? Porque a lista que vai ser ordenada será dividida em segmentos ou subpartes para proceder à sua ordenação.

Tal como o **ShellSort** tinha como conceito fundamental o

intervalo entre os elementos, o **Quick** tem como conceito chave o elemento do meio do segmento que deverá ordenar. Se considerarmos um *array* **lista** com dimensão de 10 elementos, o meio seria igual a **(10+1) div 2** (**div** representa divisão inteira em Pascal). Se estivéssemos a considerar apenas parte do *array*, o meio seria obtido pela expressão **(elem. inicial + elem. final) div 2.**

Depois de seleccionar um meio, o **Quick** coloca todos os elementos menores que o valor na posição média numa parte inferior do *array* e todos os maiores numa parte superior. Aí volta a repetir todo o processo, mas para cada uma das partes do *array*. Este vai sendo segmentado aos poucos e poucos. Tudo isto será repetido para cada parte e subparte do *array* até este estar completamente ordenado.

Tal como poderá observar no código, é utilizado o conceito recursividade, o qual consiste no chamamento de uma rotina por ela própria. Neste caso, a recursividade é útil para ordenar subpartes do *array* sempre com a mesma rotina.

A actuação desta rotina sobre o *array* **lista** com, por exemplo, 10 elementos, teria como chamamento inicial **quicksort(1,10)** no Pascal e **quicksort(0,9)** no C. A diferença acontece pois o C considera a primeira posição dum array a **0** e o Pascal é conforme definição. O normal é definir o *array* como **[1..10]**, o que origina tal chamamento.

Quadro 5 – Pascal

```
Procedure QuickSort(left,right:integer);

var media,ndx1,ndx2:integer; {variável indicadora do meio}
troca:integer;

Begin
    media:=lista[(left+right) div 2]; {obtém o valor do meio}
    ndx1:=left;
    ndx2:=right;
    repeat
        while lista[ndx1]<media do
            inc(ndx1); {incrementa ndx1 para escolha do
            segmento}
            while media<lista[ndx2] do
                dec(ndx2); {decrementa ndx2 para escolha do
                segmento}
            if ndx1<=ndx2 then
                begin
                    troca:=lista[ndx2];
                    lista[ndx2]:=troca; {troca os elementos}
                    inc(ndx1);
                    dec(ndx2);
                end;
    until ndx1>ndx2; {se ocorrer troca ndx1 e ndx2 terão novos
    valores}
    if left<ndx2 then
        quicksort(left,ndx2); {ordena apenas um segmento}
    if ndx1<right then
        quicksort(ndx1,right); {ordena apenas um segmento}
end;
```

Quadro 6 – C

```
QuickSort(left,right)

int left,right;
{
    int media,ndx1,ndx2;
    int troca;

    media=lista[(left+right) /2];
    ndx1=left;
    ndx2=right;

    do
    {
        while(lista[ndx1]<media) ndx1++;
        while(media<lista[ndx2)] ndx2- -;
        if(ndx1<»ndx2)
        {
            troca=lista[ndx1];
            lista[ndx1]=lista[ndx2];
            lista[ndx2]=troca;
            ndx1++
            ndx2- -;
        }
    while(ndx1<=ndx2);
    if(left<ndx2) quicksort(left,ndx2);
    if(ndx1<right) quicksort(ndx1,right);
}
```

NA DISQUETA

Poderá encontrar na disqueta um programa de demonstração escrito em Pascal que compara os tempos de execução dos três métodos. Não aconselho os testes com menos de 1000 elementos pois é possível que obtenham **0.00** pois o tempo é feito com uma leitura aproximada.

CONCLUSÃO

Qualquer programador que domine razoavelmente uma destas linguagens e tenha percebido as explicações do artigo, deverá conseguir adaptar estas rotinas a casos específicos. Qualquer dúvida ou problema deverá ser endereçada à **Spooler** ou a mim directamente através da morada mencionada no programa.

Não falte ao encontro marcado no próximo número onde falarei da ordenação de estruturas de dados não singulares e das ordenações em disco. Até à próxima.

Pedro Andrade

MIL FONTES

> **Uma das características que fez com que o Windows 3.0 se tornasse tão popular foi a inclusão de fontes escaláveis chamadas «TrueType».**

Não, não lhe vamos oferecer mil fontes para o seu Windows! Ao invés vamos-lhe oferecer na disqueta **Spooler** deste mês algo muito mais útil: um programa para listar todas as suas fontes.

Uma das características que fez com que o Windows 3.0 se tornasse tão popular foi a inclusão de fontes escaláveis chamadas «TrueType». De facto, até esse momento apenas os profissionais de DTP e os mais aficionados dispunham de software capaz de produzir fontes que apresentavam sempre o seu melhor no ecrã ou na impressora em qualquer tamanho.

O «TrueType» (que por mero acaso já era usado há anos em ambiente Mac, coincidências...) veio assim democratizar o panorama, permitindo ao vulgar utilizador ocasional de Windows produzir documentos com aspecto profissional.

No entanto, após usar todas as (poucas) fontes que vêm com o Windows, fica-se sempre com «fome» de mais. E nessa altura começaram a aparecer fontes vindas dos mais variados quadrantes: *packages* profissionais com largas dezenas de fontes, aparecimento em massa de fontes em BBS e mesmo programas «faça-você-mesmo» para qualquer um criar ou editar fontes escaláveis.

Ao fim das primeiras cem fontes, é frequente o coleccionador perguntar a si próprio: «Como se chama aquela fonte em que as letras aparecem como na «Guerra das Estrelas?». E nessa altura o nosso hipotético coleccionador vai procurar debalde um utilitário que lhe permita listar todas as fontes em papel ou ecrã. E de facto recebe um balde de água fria pois tal utilitário é bastante difícil de encontrar – eu sei, pois já estive nessa situação. Resta-nos ir ao «Control Panel» e ir olhando para as fontes uma a uma.

Como um colaborador da **Spooler** não se deixa derrotar por estes pequenos pormenores, toca de desenfurrejar o compilador! O resultado foi o programa PRINTFONT que imprime TODAS as fontes existentes no Windows, usando várias páginas caso tal seja necessário.

As impressões deste programa vão permitir ver claramente as diferenças entre as fontes «TrueType» e as «Bitmap». Repare nas escadas que aparecem nas letras em «System» e na perfeição com que aparece a «Times New Roman» para que se dissipem todas as dúvidas sobre o tipo das fontes! Note-se que, além dos tipos de fonte «TrueType» e «Bitmap», existem ainda as fontes de ecrã (que são tratadas como qualquer outra «Bitmap») e as de impressora. Estas últimas têm medidas fixas não sendo possível escalá-las pois ficam assim sempre preparadas para impressões com as fontes internas da impressora. Dica: use as fontes internas da impressora sempre que quiser impressões «super-hiper-rápidas».

Como algumas das fontes são símbolos, é incluído juntamente com um exemplar do tipo de letra uma pequena nota com o nome da fonte.

Por último, fica desde já o aviso que o tempo de execução deste programa pode ser de meia dúzia de segundos até vários minutos, dependendo da velocidade do seu computador e do número de fontes que tiver. Não se espante se o seu computador ficar durante minutos a mexer no disco, pois lembre-se que ele está a movimentar quantidades enormes de informação ao ler do disco, escalar, produzir o *bitmap* e imprimir cada fonte. Como nota positiva, este não é o tipo de programa que vai usar todos os dias. Pelo menos em princípio...

Permitam-me uma pequena nota: em condições exactamente iguais de CPU, memória e disco, o Windows for Workgroups é entre 10 a 20% mais rápido a gerar fontes que o Windows 3.1. Isto acontece devido ao facto de as fontes terem sido uma das áreas optimizadas para 32 Bits no Windows for Workgroups (W4WG).

O Windows/NT com 2 vezes mais memória (16Mb em vez de 8Mb) forneceu uma *performance* (leia-se «desempenho») ligeiramente inferior ao W4WG. Ou seja, nada de estranhar se nos lembrarmos que o NT tem que se preocupar com uma série de «pormenores» com que o W4WG nem sequer sonha. A componente Windows do OS/2 portou-se muitíssimo melhor que todas as versões do Windows, mas não vou revelar valores pois o teste foi feito noutra máquina e a discrepância é tão grande que posso arriscar-me a ser processado pela Microsoft! (Estou apenas a brincar, obviamente).

Dado isto, passemos então à parte técnica onde revelarei as magias e poções usadas para criar tal programa. A descrição que se segue parte do princípio que o leitor já sabe algo de programação em Windows, pelo que convém fazer a habitual ressalva para quem não pertencer a esta categoria: passe a ler o último parágrafo pois os escritos que se seguem contêm frases altamente chocantes!

A MAGIA DO CALLBACK

A chave que nos permite abrir todas as portas de criação de fontes é uma função obscura do extenso API do Windows. Trata-se do **EnumFonts**. Esta função recebe como parâmetros:

– O «Display Context» do dispositivo para o qual se quer gerar a fonte. Pode ser qualquer dispositivo de impressão ou placa de vídeo suportada pelo Windows. No nosso caso trata-se da impressora escolhida pelo utilizador.

– O Nome da fonte a gerar. Como nós pretendemos que o EnumFonts nos gere todas as fontes, basta assumir «NULL» para este parâmetro.

– A rotina de «callback» (quem conseguir uma tradução decentemente curta para Português deste termo tem direito a uma menção honrosa no próximo número da **Spooler**!). Mas afinal o que é uma rotina *callback*? É uma rotina que é invocada **n** vezes como resultado da iteração dentro de outra rotina. Complicado? Para descomplicar, vou esclarecer o que se passa no nosso caso: a função EnumFonts vai invocar a função que é passada como parâmetro **n** vezes, tantas quantas as fontes existentes. E claro que serão passados a essa rotina certos parâmetros segundo uma ordem pré-estabelecida. Como isto é motivo de extenso tratamento em qualquer manual de Windows, vou deixar o leitor encarregar-se de pesquisar mais a fundo esta característica.

– Um parâmetro adicional a passar aquando da invocação da nossa rotina de *callback*. No nosso caso vamos usar este campo a «NULL», pois não vamos precisar de mais nenhuma informação a não ser o nome da fonte.

Depois de dominar esta função bastante potente, o resto do trabalho fica simplificado. É muito importante notar que todas as rotinas *callback* devem ser instanciadas através do **MakeProcInstance** e posteriormente libertada a memória através de **FreeProcInstance**. Sem entrar em grandes pormenores, isto verifica-se pois estas funções são exportadas. Para já o leitor deve aceitar isto como dogma pois a explicação vai cair no campo puro do *Assembly*, que é um tema desconhecido para muitos. E fazem muito bem!

Olhando concretamente para as fontes do programa (o quê? ainda não as imprimiu?), este é composto pelas seguintes rotinas:

– **WndProc**: trata-se da rotina de tratamento de mensagens da nossa janelinha que, neste caso, nunca passará de um ícone. Note-se que o programa poderia ter sido feito sem sequer abrir uma janela, mas isso seria contra todas as regras de boa programação. E uma vez temos que aceitar mais este dogma: qualquer programa de Windows tem que abrir uma janela, mesmo que esta seja invisível.

– **WinMain**: o tradicional ponto de entrada do programa. Regista a classe da nossa janela, abre-a e pouco mais...

– **Header**: imprime o cabeçalho de cada página

– **EnumerateFont**: a nossa rotina de *callback*. Imprime uma amostra de cada fonte tendo o cuidado de escalar correctamente a fonte de forma a não alterar o seu *aspect-ratio*. Se for caso disso, invoca o *Header* sempre que encher uma página.

– **GetFontInfo**: invoca o *EnumFonts*, tendo o cuidado de avisar o utilizador que o processo iniciado pode demorar algum tempo através da mudança do cursor para uma ampulheta. No fim da enumeração, voltamos ao velho cursor da seta.

– **Print**: pergunta ao utilizador qual a impressora a usar, podendo este inclusivamente configurá-la a seu bel-prazer. Depois disso é criado um *Display Context* para a impressora escolhida, é aberto um ficheiro na Spooler (do Windows!) e passa-se de imediato à impressão. No fim de tudo, faz as habituais tarefas de limpeza da casa: recuperar a memória alocada, fechar o documento na Spooler do Windows, etc...

Quanto à parte técnica, nada mais existe para acrescentar. Aconselho a consulta da fonte do programa pois está comentada de forma a explicar os passos mais obscuros.

ANTES DE ME IR EMBORA...

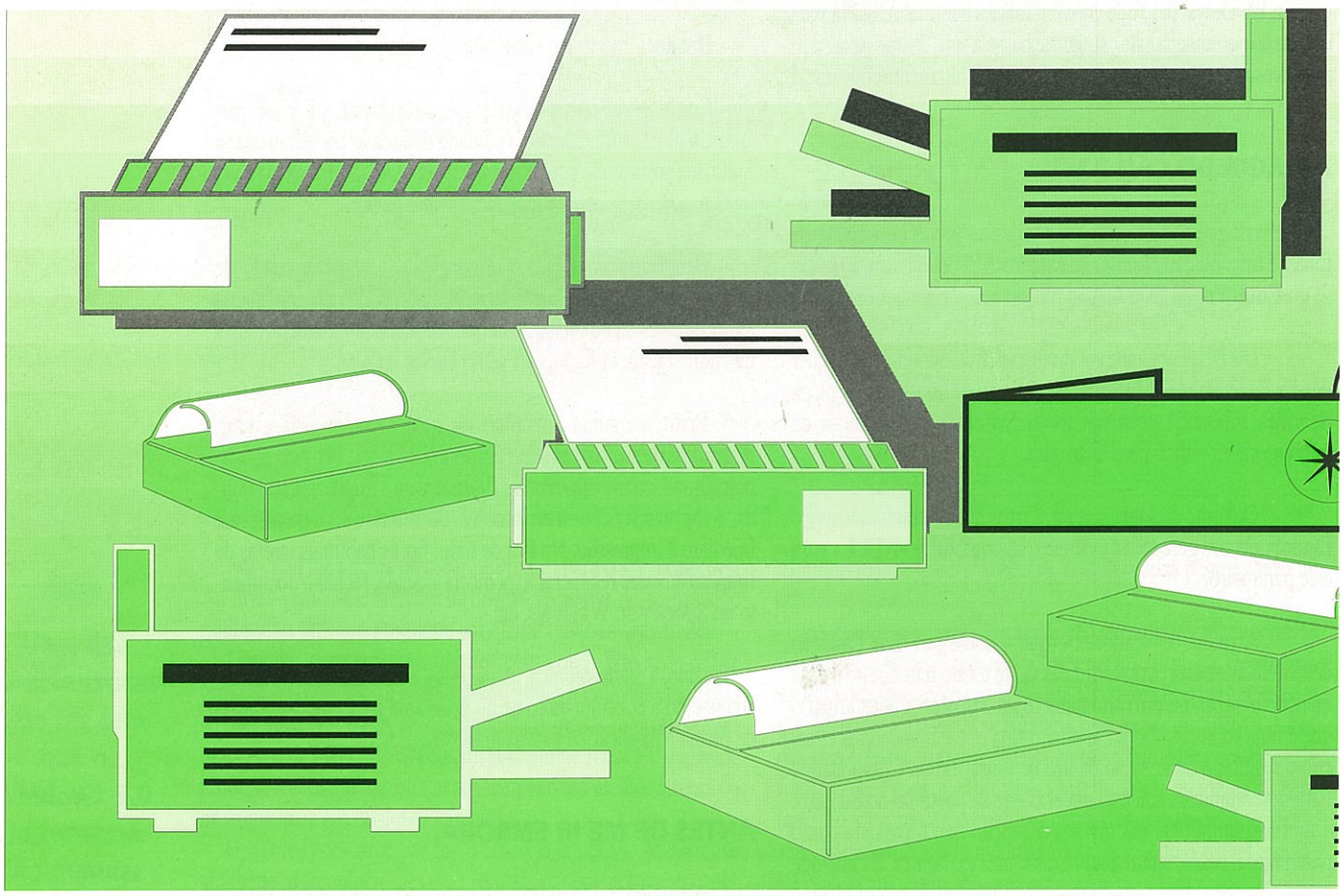
... permitam-me contar a minha própria experiência antes e depois de criar este programa: tinha o meu disco atafulhado com mais de 200 fontes mas nunca devo ter usado mais de 10% desse valor, pois é bastante penoso estar a pesquisar entre tantas fontes a indicada para determinado trabalho. Depois do PRINTFONT... tudo mudou pois as tais 200 e tais fontes resumiram-se a duas míseras folhas de papel que tenho sempre junto ao computador. Não sei se será impressão minha, mas creio que desde há pouco tempo os meus documentos estão a sair com um aspecto gráfico melhor. (Pelo menos eu estou convencido disso, o que já é alguma coisa...).

No caso do leitor, já não tem desculpa para não começar a sua colecção de fontes! E quem sabe se a **Spooler** oferecerá em números futuros uma colecção de fontes de forma a ajudar os leitores a juntarem as suas primeiras **MIL FONTES**?

Miguel Vitorino

IMPRESSORAS

PARTE VI



Continuando a sequência de artigos sobre impressoras, vamos desta vez abordar o modo **HP-LaserJet**. Mais uma vez e à semelhança do que se passou com os modos EPSON e IBM, torna-se necessário dividir o artigo devido à sua extensão. Assim, nesta primeira parte irão ser abordados os comandos mais genéricos, deixando para a segunda as fontes e os gráficos.

NOTA: Os parâmetros numéricos que alguns comandos recebem, não são valores ASCII mas sim sequências de dígitos (0..9). Estes podem ainda incluir sinais (+ e -). Segue-se um exemplo: -234

CÓDIGOS DE CONTROLO

BS	8

BackSpace

Move a posição de impressão um espaço para a esquerda. O espaço é baseado na largura da fonte de caracteres actual. Se a posição de impressão já estiver o mais à esquerda possível, este comando é ignorado.

LF	10

Line Feed

Avança o papel uma linha, mantendo a posição de impressão na mesma coluna.

FF	12

Form Feed
Avança o papel para o topo da próxima página.

CR	13

Retorno do carro
Retoma a posição de impressão à margem esquerda, mantendo a mesma linha.

SO	14

Shift Out
Selecciona os caracteres da fonte secundaria até que seja detectado o comando SI.

SI	15

Shift In
Selecciona os caracteres da fonte principal até que seja detectado o comando SO.

FORMATAÇÃO DA PÁGINA

ESC & l # P	27 38 108 # 80

Especifica o comprimento de página
Altera o tamanho da página para # linhas, utilizando o espaçamento entre linhas corrente.

ESC & l # E	27 38 108 # 69

Define a margem de topo
Altera o tamanho da margem de topo para # linhas, utilizando o espaçamento entre linhas actual.

ESC & l # F	27 38 108 # 70

Selecciona o tamanho do texto
O tamanho do texto é definido como sendo o tamanho da página menos as margens de topo e de fundo. O parâmetro # determina o número de linhas de texto por página.

ESC & l # D	27 38 108 # 68

Define o Nº de linhas por polegada
Os incrementos verticais são definidos em linhas por polegada pelo parâmetro #. Este pode conter os seguintes valores: 1, 2, 4, 6, 8, 12, 16, 24, 48.

ESC & l # C	27 38 108 # 67

Vertical motion index
O parâmetro # especifica o número de 1/48 polegadas a utilizar no espaçamento de uma linha. Valores superiores a 126 (para *n*), serão ignorados.

ESC & k # H	27 38 107 # 72

Espaçamento horizontal de impressão
O parâmetro # especifica o número de 1/120 polegadas a utilizar no espaçamento horizontal. O valor do parâmetro # pode variar entre 0 e 126.

ESC & a # L	27 38 97 # 76

Define a margem esquerda
Define a margem esquerda como sendo a coluna colocada em #, se este valor for superior à posição da margem direita, o comando será ignorado.

ESC & a # M	27 38 97 # 77

Define a margem direita
Define a margem direita como sendo a coluna colocada em #, se este valor for inferior ao da margem esquerda, o comando será ignorado.

ESC 9	27 57

Apaga margens laterais
Este comando faz um *reset* às margens laterais, colocando-lhes os seus valores iniciais.

ESC & a # C	27 38 97 # 67

Posição de impressão horizontal
Este comando move a posição de impressão para a coluna #, utilizando a largura da fonte actual. A posição de impressão pode tanto ser movimentada para a esquerda como para a direita, utilizando respectivamente números negativos ou positivos.

ESC & a # R	27 38 97 # 82

Posição de impressão vertical
Este comando posiciona a posição de impressão numa linha especificada, permanecendo na mesma coluna. O parâmetro #, especifica o número relativo de linhas a movimentar na vertical, para cima ou para baixo.

ESC & a # H	27 38 97 # 72

Posição de impressão horizontal (pontos)
Este comando move a posição de impressão horizontalmente em incrementos de 1/720 polegada. Este comando ignora margens, sendo o único limite a folha de papel. O parâmetro # pode tanto especificar uma movimentação para a esquerda bem como para a direita, recorrendo respectivamente à utilização de números negativos ou positivos.

ESC & a # V	27 38 97 # 86

Posição de impressão vertical (pontos)
Este comando move a posição de impressão verticalmente em incrementos de 1/720 polegada. Este comando ignora margens, sendo o único limite a folha de papel. O parâmetro # pode tanto especificar uma movimentação para cima bem como para baixo, recorrendo respectivamente à utilização de números negativos ou positivos.

COMANDOS GENÉRICOS

ESC & p # X	27 38 112 # 88

Impressão transparente de dados
Este comando permite enviar códigos de controlo para a impressora sem que esta os interprete. O parâmetro # indica o

número de bytes que seguem o X que deverão ser ignorados, isto é, impressos.

ESC & l # L	27 38 108 # 76

Performation skip mode

Este comando controla a utilização, ou não, da margem inferior da página. O controlo é efectuado pelos valores passados no parâmetro #, 0 ou 1.

ESC & d D	27 38 100 68

Sublinhado

Todos os caracteres depois deste comando serão sublinhados.

ESC & d @	27 38 100 64

Cancela sublinhado

Cancela a impressão em sublinhado

ESC & k # G	27 38 107 # 71

Terminação de linha

Este comando determina a forma como a impressora deverá detectar o fim de linha. O valor do parâmetro # está descrito na tabela que se segue, sendo o seu valor por omissão, 0.

MODO (#)	DEFINIÇÃO
0	CR -> CR , LF -> LF , FF -> FF
1	CR -> CR+LF , LF -> LF , FF -> FF
2	CR -> CR , LF -> CR+LF , FF -> CR+FF
3	CR -> CR+LF , LF -> CR+LF , FF -> CR+FF

ESC & s # C	27 38 115 # 67

Modo wrap

Este modo faz com que todos os caracteres que passarem a margem direita de uma linha sejam ou não cortados e transportados para a linha seguinte. Se o parâmetro # for 0 este modo é inibido, se for 1 este modo é desinibido.

ESC & l # X	27 38 108 # 88

Selecciona o número de cópias

Este comando permite definir o número de cópias através do parâmetro #. Este pode variar entre 1 e 99.

ESC & l # H	27 38 108 # 72

Controlo da entrada de papel

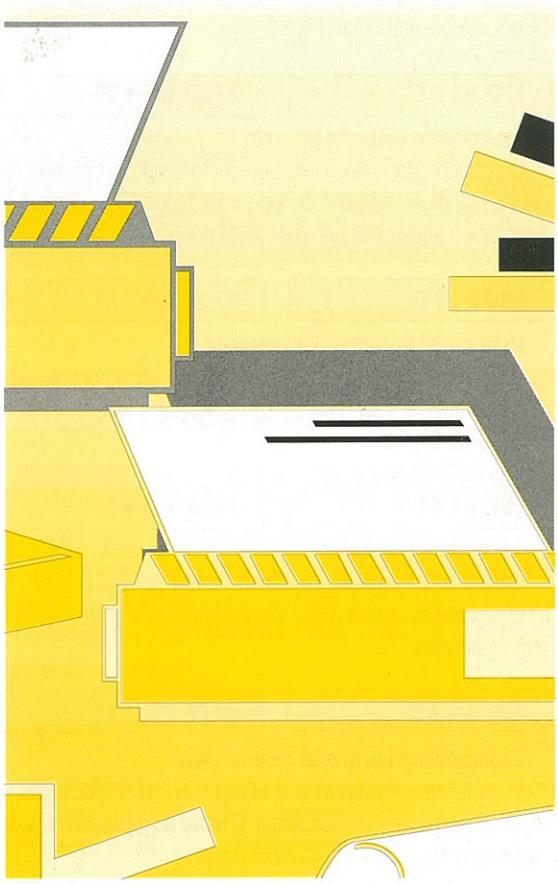
Este comando permite controlar a forma como o papel é introduzido na impressora. Pode-se visualizar a correspondência entre o parâmetro # e o método na tabela seguinte:

VALOR (#)	MODO
0	Ejecta a página
1	Selecciona o hooper 2
2	Selecciona o hooper 2
3	Selecciona o hooper 1
4	Selecciona o hooper 1

PARA A PRÓXIMA

No próximo número iremos concluir a abordagem do modo HP-LaserJet, onde serão descritos temas de grande importância, nomeadamente fontes e gráficos. A complementar tudo isto serão dados alguns exemplos elucidativos das funções mais importantes deste modo, como habitual, em linguagem C.

Carlos Ladeira



ShaRe WARE

PARTE II

Os programas de telecomunicações no «mercado» de *shareware* sempre foram abundantes, sendo até bastante difícil de encontrar algum que não tivesse um mínimo de qualidade, mas o que seria bastante difícil de encontrar, senão mesmo impossível, seria um programa de comunicações em português. Essa lacuna veio a ser preenchida pelo Lusocomm, um programa de comunicações em português, cujo autor é já bastante conhecido no mundo das BBS Portuguesas: Pedro Duarte. Autor esse que vai ser entrevistado em exclusivo para a **Spooler**, onde vamos tentar «desvendar» porque é que ele fez uma aposta que se poderia considerar falhada desde o início, não porque o programa tenha falta de qualidade – porque a tem – é um programa muito bom que poderia competir com os melhores do mercado se fosse feito noutro idioma. Mas não é nossa intenção criticar os actos do Pedro Duarte, antes pelo contrário: essa sua atitude é de LOUVAR, visto que ele fez uma aposta ousada que era de fazer um programa em comunicações. No fim desta entrevista esperamos saber se realmente valeu a pena. O mercado de *shareware* em Portugal está no começo do seu desenvolvimento e, se as pessoas não correspondem e não registam os programas que usam, o mercado acabará por não ter sucesso e consequentemente acabará por morrer ainda «antes de começar», o que seria muito triste. Mas é mesmo essa umas das intenções desta coluna na **Spooler**, dedicada ao *shareware* português: incentivar as pessoas não só a registar o *software* de qualidade, mas também a escrever e lançar nesse mesmo mercado os seus programas. Mas comecemos então com a entrevista:

P1: Porque é que decidiste escrever o Lusocomm?

Apenas porque me interessei pela programação relacionada com comunicações. Quando comecei a fazer o LusoComm, os meus conhecimentos nesta área de programação eram nulos e têm vindo a evoluir à medida que o programa avança.

P2: Porquê escrever um programa em português, quando é mais que sabido que isso não iria compensar o esforço, não só devido à pequenez do mercado português, como também ao mau hábito dos portugueses de não respeitarem os autores de qualquer tipo de *software* (*shareware* ou não)?

Precisamente pelo facto de que não existia nenhum programa deste tipo em português (pelo menos que eu tivesse conhecimento). Já existiam dezenas de programas de comunicações em inglês, e pareceu-me ser bem mais útil, original e estimulante fazer um na minha própria língua.

P3: Para quando o fim das versões Betas e o lançamento da primeira versão «oficial»?

Ultimamente, por motivos vários, não tenho dedicado muito tempo ao programa, pelo que me é impossível dar uma resposta concreta.

P4: Para quando uma versão internacional, em inglês?

Não posso avançar datas exactas, mas reafirmo que sempre foi minha intenção fazer uma versão do programa em inglês (ou, melhor ainda, multi-língua).

P5: Quais as inovações que ainda pretendes introduzir no teu programa?

Menus de ajuda sensível ao contexto, melhorar a linguagem «script» e protocolos internos. Há mais coisas, mas estas são talvez as mais importantes.

P6: Achas que valeu o esforço, isto é os utilizadores tem correspondido?

Mesmo que nenhum utilizador tivesse registado o LusoComm, no fundo eu não estaria de modo algum arrependido de o ter feito. Durante muito tempo nunca pensei que o programa pudesse chegar ao ponto a que chegou, e pode-se até dizer que, pelo menos até uma certa altura, encarei todo este projecto mais como um

"Os programas de telecomunicações no «mercado» de *shareware* sempre foram abundantes, sendo até bastante difícil de encontrar algum que não tivesse um mínimo de qualidade"

«hobby» do que outra coisa qualquer. Mas sem dúvida que o facto de os utilizadores terem correspondido bastante bem também foi muito gratificante.

P7: Que papel tiveram as BBS no desenvolvimento do programa, não em termos de o objectivo do programa ser as BBS, mas em ajuda, testes, etc? Achas positivo? Porquê?

Tiveram um papel decisivo! Se não fossem as BBS, muito provavelmente o LusoComm não existiria. Na fase inicial do projecto, com a preciosa ajuda do Sr. Paulo Gradim (ex-coordenador da rede FreeNet, agora LusoNet), consegui reunir uma equipa de «beta-testers» que, através do correio entre BBS, teve um papel muito importante nas primeiras versões do programa. Depois da primeira versão pública, a criação de uma conferência de suporte ao LusoComm também tem sido uma óptima experiência, com resultados muito positivos.

P8: Que achas do *Shareware* Nacional pré-LusoComm e pós-Lusocomm? Ou melhor, o que achas do resto do *Shareware* nacional existente?

Existe pouca coisa, mas o que existe é bom! Não posso deixar de mencionar o primeiro programa de Shareware nacional que me deixou deveras impressionado, e que ainda hoje considero indispensável: o excelente V860, da autoria do Sr. José Páscoa.

P9: O que é que achas que é preciso para que o mercado de *shareware* evolua em Portugal?

Como para muitas outras coisas, é necessário uma alteração das mentalidades. Só que, para conseguir tal coisa, é essencial que os preços praticados estejam de acordo com a realidade portuguesa. É óbvio que isto não se aplica apenas ao shareware, mas também e principalmente, ao software comercial «normal».

P10: O *shareware* têm futuro como meio de distribuição? E em Portugal com as suas características peculiares no que toca ao uso de *software*?

Eu não tenho qualquer dúvida que o shareware é um excelente meio de distribuição. Há autores nos EUA que vivem dos registos obtidos com os seus programas shareware. Embora para a maioria a situação seja bem diferente, é inquestionável que o shareware é o melhor (talvez o único) método de comercializar programas que de outro modo não teriam qualquer difusão. E não se pense que isto significa que os programas shareware não têm qualidade, pois a realidade prova precisamente o contrário.

Em relação à situação portuguesa, o shareware é apenas a «ponta do iceberg». Como já referi, há que alterar as mentalidades em relação ao software comercial, mas é preciso criar bases sólidas para que isso aconteça. O shareware virá por acréscimo. Por outro lado, também se poderá argumentar que, devido ao seu baixo preço, talvez as pessoas comecem por registar o shareware, mas parece-me uma visão um pouco utópica.

P11: Dá alguns conselhos aos leitores que desejem lançar os seus programas no mercado de *shareware*.

É importante fazer um esforço para que todo o pacote (programa, documentação, etc.) tenha o melhor aspecto possível, com um nível de qualidade a condizer, de forma a transmitir confiança aos utilizadores. Também é cada vez mais comum os programas preparados de origem para suportar várias línguas, o que me parece ser uma excelente «feature». Outra coisa importante é facilitar ao máximo o sistema de pagamento (se possível, através de cartões de crédito).

Gostaria de terminar agradecendo a todos aqueles que contribuiram para tornar esta entrevista possível, nomeadamente aos responsáveis por esta coluna na revista ***Spooler Magazine****: o Tiago Pascoal e o Carlos Batista.*

Este ponto de encontro sobre *shareware*, fica assim este mês exclusivamente dedicado ao inquestionavelmente melhor programa de *shareware* português do momento. Mais uma vez os nossos agradecimentos ao Pedro Duarte por se ter prestado a esta entrevista pública e os esclarecimentos prestados por alguém que «vive e sente» o *Shareware* Português. Foi a justa homenagem ao, como já dissemos, melhor programa de *shareware* português do momento.

Em relação ao conteúdo da entrevista em si, ela vem provar que nem só os responsáveis desta coluna têm a opinião de que em Portugal, o principal problema do *shareware* é uma «crise de mentalidade», quer sobre a maneira de ver o *shareware*, quer sobre a maneira de ver o esforço e trabalho do programador, qualquer que ele seja (ver artigo do Número 26 da **Spooler**).

Em relação ao Lusocomm como programa, devemos acrescentar a tudo o que já se leu sobre ele, de que se necessita de um programa de comunicações fácil de configurar, pequeno e que faça as principais (e não só...) funções de um programa desse tipo, então o que procura é o Lusocomm. Sublinhando que a sua principal «feature», para nós, é a sua facilidade de configuração, mesmo para aqueles que iniciam o contacto com um programa do género pela primeira vez. Mas o melhor, talvez, é ver pelos seus próprios olhos o Lusocomm, ele encontra-se disponível nas disquetas de *shareware* da «Montra Spooler».

Para o mês esperamos contar com uma mini-entrevista a outro promissor programador de *shareware* da nossa «praça», assim como inaugurar um canto deste artigo, onde vamos dedicar especial atenção às notícias mais relevantes sobre *shareware* e afins ocorridas nesse mesmo mês.

Carlos Batista e Tiago Pascoal

MEMÓRIA ESTENDIDA

PARTE IV

O Modo Protegido de trabalho do processador Intel de 32 bits foi idealizado, concebido e desenvolvido pelos engenheiros da Intel com um importante objectivo em vista: dar um bom suporte a nível do «hardware» do microprocessador às actividades de **Multitarefa** (ou «Multitasking» no seu nome de origem), isto é, a execução simultânea de diferentes programas ou processos.

Pretendia-se com isso acompanhar a linha de evolução previsível e desejável para os novos sistemas operativos e, aqui, o pensamento estava certamente dirigido (nessa altura, claro) para o OS/2 da IBM que vinha desde há muito sendo anunciado como o futuro grande sucessor do DOS.

Como os leitores estarão eventualmente lembrados, o OS/2 tivera um ínicio de carreira pouco fulgurante, conduzindo a um desastroso insucesso comercial na opinião de alguns. A generalidade dos utilizadores não emigrou para o OS/2 como esperado e continuou calmamente apegado ao velho DOS.

Houve que fazer então um apuramento de responsabilidades para essa estranha situação e, talvez sem grande surpresa, concluiu-se que o maior culpado era o microprocessador 80286, em particular pelo seu fraco apoio às actividades de **Multitarefa.** Isto porque nos microprocessadores 80286 cada segmento de memória estando limitado a um máximo de 64 KB, não é possível endereçar mais de 16 MB de memória e a compatibilização do **Modo Protegido** com o **Modo Real** é verdadeiramente problemática.

A resposta da Intel a essas questões veio com os microprocessadores de 32 bits e foi nada mais nada menos que:

a) Levantamento das restrições ao tamanho dos segmentos de memória. A partir de então um segmento de memória poderá ter até 4 Gigabytes de tamanho.

b) A memória endereçável poderá existir ou não fisicamente em RAM. O microprocessador dispõe de um mecanismo de *Paging* que, se for utilizado, permite mapear endereços lineares para endereços físicos completamente distintos. Deste modo é possível, entre outras coisas e só a título de exemplo, manter parte da informação de um programa (código ou dados) no disco rígido, carregá-la para uma zona tampão da memória física, quando for necessário, mapear quaisquer endereços lineares para esses endereços de memória física e fazer prosseguir a execução do programa normalmente. E quando essa informação deixar de ser necessária, ela pode ser descarregada de novo para o disco rígido.

Naturalmente que consecutivos acessos ao disco rígido tornam o sistema significativamente mais lento, mas também não se pode ter tudo e 1 MB de RAM é muito mais caro que 1 MB de espaço em disco rígido.

c) Um modo de emulação do **Modo Real** de funcionamento dos processadores 8086, mas compatível com o **Modo Protegido** e a gestão de memória em **Modo Protegido**.

Este modo de trabalho é designado por **Modo Virtual 86**.

Programas como o EMM386.EXE o QEMM386.SYS e o 386MAX.SYS colocam o processador em **Modo Virtual 86** e utilizam o mecanismo de *Paging* do **Modo Protegido** para mapear RAM nas zonas de memória conhecidas por **UMB**s (*Upper Memory Blocks*). As janelas de DOS no Windows e no OS/2 também são obtidas colocando o processador a trabalhar em **Modo Virtual 86**.

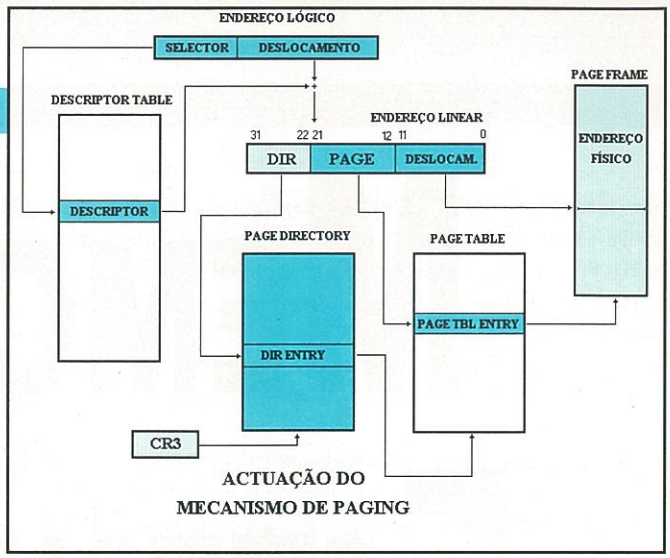
Neste nosso artigo vamos então abordar (infelizmente de modo sumário) dois tópicos extremanente importantes para complementar o estudo que temos vindo a realizar sobre o **Modo Protegido**:

- O mecanismo de *Paging*.
- O suporte dado pelos processadores de 32 bits às actividades de **Multitarefa**.

Vamos ainda ficar a saber o que são as *Call Gates*, e porque existe necessidade de mais este «portão» para que todo o modelo do **Modo Protegido** funcione sem atritos.

Para o próximo número da revista apresentaremos o programa VIRT86.EXE, muito interessante e didático em nossa opinião, e que ilustrará amplamente todos os temas da matéria coberta ao longo dos vários artigos sobre as bases fundamentais do **Modo**

Vamos ainda ficar a saber o que são as *Call Gates*, e porque existe necessidade de mais este «portão» para que todo o modelo do Modo Protegido **funcione sem atritos.**



ACTUAÇÃO DO MECANISMO DE PAGING

Protegido, e ainda a faceta mais enigmática e incompreendida do funcionamento do processador Intel de 32 bits, o **Modo Virtual 86**, irá (nunca tal vimos até hoje) ser integralmente divulgada em público através do código fonte de um programa.

Mas até lá recomendamos um bom estudo de linguagem *Assembly*, pois creiam-me, não obstante o muito que se diz por aí, vale mesmo a pena.

O MECANISMO DE PAGING

Quando o mecanismo de *Paging* é utilizado, a correspondência endereço linear– endereço físico é feita através de tabelas colocadas em memória pelo programa supervisor do Modo Protegido. Essas tabelas designam-se por *Page Tables* e gerem «talhadinhas» de 4 KB de memória designadas por *Pages*.

Existem 2 níveis de *Page Tables*, o *Page Directory* e as *Page Tables* de 2º nível (muitas vezes designadas simplesmente por *Page Tables* quando não houver possibilidade de permanecer qualquer confusão).

Tanto um como outro dos níveis de *Page Tables* é constituído internamente por entradas de 4 bytes designadas por *Page Table Entries* e que apresentam o formato da figura seguinte.

P - Segmento presente em memória /Não presente (1 / 0)
R/W - Leitura / Escrita (0 / 1) U/S - Utilizador / Supervisor (1 / 0)
A - Acedido /Não acedido (1 / 0) D - "Dirty" / "Clean" (1 / 0)
AVL - Disponível / Indisponível para o programa supervisor

Entrada no "Page Directory" ou na "Page Table" de 2º nível.

O mecanismo de *Paging* considera um endereço linear de 32 bits como sendo constituído por 3 partes:

– Os bits 22 até 31 (últimos 10 bits) do endereço linear constituem um índice relativamente ao endereço físico base de um *Page Directory*.

– Os bits 12 a 21 do endereço línear constituem um índice relativamente ao endereço físico base de uma *Page Table de 2º nível*.

– Finalmente, os bits 0 a 11 constituem o valor de deslocamento (ou *offset*) na página de memória definida na entrada da *Page Table de 2º nível* obtida através dos restantes bits do endereço linear.

Por sua vez o valor do endereço físico base (valor de 32 bits) do *Page Directory* consta de um registo do microprocessador designado por CR3, ou PDBR (*Page Directory Base Register*).

Os endereços físicos base quer do *Page Directory*, quer das *Page Tables de 2º nível* quer das *Pages* de memória são sempre localizados em endereços físicos de memória múltiplos de 4 KB. Repare agora na figura seguinte pois talvez ajude bastante na consolidação de ideias:

Quando o microprocessador descodifica um endereço linear, procede do seguinte modo:

– O valor correspondente aos bits 22 a 31 do endereço linear é multiplicado por 4 (número de bytes numa *Page Table Entry*) e adicionado ao valor existente no registo CR3. Encontra assim o endereço físico de uma entrada no *Page Directory*. Nessa entrada do *Page Directory* os bits 12 a 31 (zona designada por Endereço Físico do *Page Frame* ou mais vulgarmente e no original por *Page Frame Adress*) são os 20 bits superiores do endereço físico base de uma *Page Table de 2º nível*.

Como se disse atrás, os endereços base destas tabelas situam-se em endereços múltiplos de 4KB ou 2^{12} bytes, por conseguinte os bits 0 a 11 do endereço físico têm sempre o valor 0, não havendo portanto qualquer ambiguidade na determinação do endereço físico.

– Seguidamente o valor correspondente aos bits 12 a 21 do endereço linear é multiplicado por 4 e adicionado ao endereço físico base da *Page Table de 2º nível*.

O microprocessador obtém deste modo o endereço físico de uma entrada na *Page Table de 2º nível*.

– Nessa entrada da *Page Table de 2º nível* o *Page Frame Adress* representa os 20 bits superiores do endereço físico base de um *Page Frame* de memória.

Os 12 bits inferiores são sempre 0, conforme se referiu acima.

– Finalmente os bits 0 a 11 do endereço linear são adicionados ao endereço físico base do *Page Frame* de memória, ficando assim completada a descodificação de endereço linear para endereço físico.

Repare agora de novo na figura representativa de uma *Page Table Entry* e nos vários campos que aí aparecem para além do endereço físico do *Page Frame*:

– Campo **P** (**Present** bit): Informa se o *Page Frame Adress* mapeia para uma *Page* existente em memória física. Se o campo **P** for 0 numa *Page Table Entry* de qualquer das *Page Tables*, a tentativa do processador para utilizar essa *Page Table Entry* para descodificar um endereço dará origem a uma *Page-Fault Exception*.

Essa *Exception* possibilitará a um sistema que utilize memória virtual em disco, copiar a *Page* em falta do disco para a memória física, actualizar a informação na *Page Table Entry* e fazer prosseguir o programa como se nada se tivesse passado.

– Campo **R/W** (**Read/Write** bit) informa se o acesso à *Page* definida pela *Page Table Entry* é só de leitura ou pode ser tanto de leitura como de escrita.

– Campo **U/S** (**User/Supervisor** bit). Algumas *Pages* são para

utilização exclusiva do Sistema Operativo ou programas supervisores do **Modo Protegido**. Se o CPL (*Current Privilege Level*) for 0, 1 ou 2 o processador encontra-se a executar em modo Supervisor (do ponto de vista do mecanismo de *Paging*) e pode utilizar *Pages* definidas por *Page Table Entries* com o valor 0 no campo U/S. Se o CPL for 3 então só são utilizáveis *Pages* definidas por *Page Table Entries* com o valor 1 no campo U/S.

– Campo **A** (**Acessed** bit). Como o nome aparenta dizer, este campo informa se uma *Page* ou *Page Table de 2º nível* foi acedida em leitura e/ou escrita.

– Campo **D** (**Dirty** bit) informa se uma *Page* foi acedida para escrita.

– Campo **AVL** (**Available** bits) está livre para a utilização que for julgada mais conveniente pelos programadores do sistema.

A activação do mecanismo de *Paging* requer como mínimo a seguinte sequência de acontecimentos:

1) O sistema gestor do **Modo Protegido** criou pelo menos duas *Page Tables*, ou seja um *Page Directory* e uma *Page Table de 2º nível*.
2) O registo do processador CR3 é carregado com o endereço físico base do *Page Directory*.
3) O processador encontrando-se já em Modo Protegido, o bit 31 do registo CR0 do processador é carregado com o valor 1, o que pode ser conseguido com as seguintes instruções *Assembly*:

```
MOV EAX, CR0
OR EAX, 8000000H
MOV CR0, EAX
```

E é tudo o que de mais fundamental interessa conhecer sobre o mecanismo de *Paging*. Para uma aplicação e consequente aprofundamento destes conhecimentos, tente analizar muito cuidadamente o código fonte do programa VIRT86.EXE que apresentaremos no próximo número da nossa revista.

A MULTITAREFA

Existem dois tipos possíveis de *descriptors* nas *Descriptor Tables* GDT (*Global Descriptor Table*) ou LDT (*Local Descriptor Table*) que dão apoio ao mecanismo de Multitarefa.

O primeiro designa-se por *Task State Segment Descriptor* ou abreviadamente por *TSS Descriptor*, e o segundo designa-se por *Task Gate Descriptor*.

Quando um *interrupt*, *exception*, instrução *assembly* JMP ou CALL passa a execução de um programa para um *TSS Descriptor* ou *Task Gate Descriptor* dá-se aquilo que se designa por «Mudança de Tarefa» (ou talvez mais vulgarmente por *Task Switch*).

Durante um *Task Switch* o conteúdo de quase todos os registos do processador da Tarefa anterior são automáticamente guardados pelo processador numa estrutura em memória designada por *Task State Segment* (ou «TSS») antes da execução poder ser iniciada no ambiente da nova Tarefa. E quando a execução dessa nova Tarefa tiver terminado, o retorno à Tarefa anteriormente interrompida pode ser invocada com a simples execução de uma instrução *assembly* IRET. A figura seguinte mostra-nos o formato dos primeiros 104 bytes de um TSS

31	15	0	
'I/O MAP BASE'	0000000000000000	T	64
0000000000000000	LDT		60
0000000000000000	GS		5C
0000000000000000	FS		58
0000000000000000	DS		54
0000000000000000	SS		50
0000000000000000	CS		4C
0000000000000000	ES		48
EDI			44
ESI			40
EBP			3C
ESP			38
EBX			34
EDX			30
ECX			2C
EAX			28
EFLAGS			24
EIP			20
CR3			1C
0000000000000000	SS2		18
ESP2			14
0000000000000000	SS1		10
ESP1			0C
0000000000000000	SS0		08
ESP0			04
0000000000000000	'BACK LINK' PARA TSS ANTERIOR		00

TASK STATE SEGMENT

Nem todos os campos do TSS são dinâmicamente alterados durante o *Task Switch*, alguns são estáticos e apenas lidos pelo processador nessa ocasião:

– Campo **LDT**, o qual contém o «selector» da *Local Descriptor Table* da nova Tarefa.

– Os endereços lógicos dos *stacks* para os níveis de privilégio 0, 1 e 2. O modelo de **Modo Protegido** impõe que todos os anéis de protecção de nível superior (número cardinal inferior) a 3, devem dispor de *stacks* próprios para receber chamadas de níveis inferiores. Deste modo evita-se a possibilidade de um nível superior poder ser danificado porque o *stack* que lhe é fornecido por um nível inferior teve a infelicidade de ser pequeno demais.

– O bit **T** (*Trap bit*). Se este bit tiver o valor 1, o processador accionará uma *Debug Exception* sempre que se der o *Task Switch*.

– O endereço base do Mapa de Permissão de I/O (*I/O Permission Bitmap*). Este Mapa, no caso de existir, faz parte do TSS e é localizado algures acima do byte 67H.

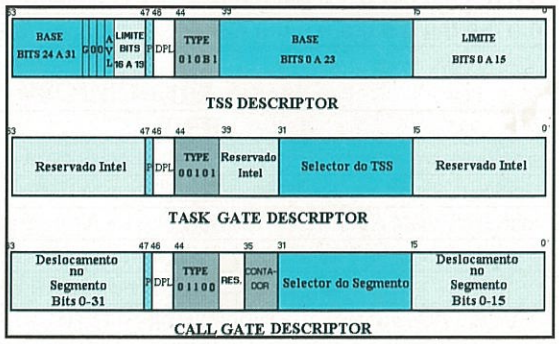
O seu endereço base é o deslocamento desde o ínicio do TSS, sendo o valor mínimo por conseguinte 68H. O valor máximo está estabelecido em 0DFFFH. O último byte do Mapa deverá ter o valor 0FFH. Com o auxílio deste Mapa o processador pode gerar *exceptions* sempre que houver acesso a determinados portos de Entrada/Saída, permitindo por conseguinte o sistema Supervisor emular ou aceder por si próprio a esses portos. Os portos que geram *exceptions* são especificados no *I/O Permission Bitmap* com o valor 1 no bit respectivo.

No programa VIRT86.EXE ilustraremos também este assunto numa perspectiva em que todo e qualquer programa irá ficar «iludido» sobre a quantidade de memória *extended* existente no

sistema enquanto o VIRT86.EXE estiver em Supervisor.

– No caso de se estar a utilizar o mecanismo de *Paging* o campo CR3 do TSS pode ser utilizado para dar a cada Tarefa as suas próprias *Page Tables*.

Vejamos agora a figura seguinte e fixemos a nossa atenção no formato de um *TSS Descriptor*.



Verifique que o campo Type apresenta um bit **B**, designado por *Busy bit*. Se o *Busy bit* tiver o valor 1, a Tarefa encontra-se em execução. As Tarefas não são **reentrantes**, qualquer tentativa de executar uma tarefa com o *Busy bit* ligado dará origem a uma *exception*.

Em Multitarefa um dos registos do processador, o *Task Register*, contém sempre o «selector» do *TSS Descriptor* corrente na GDT. Por este motivo os *TSS Descriptors* só podem existir na GDT e nunca numa LDT ou numa IDT. Para obviar a esse inconveniente utilizam-se algumas vezes *Task Gates*. Aqui o campo designado por Selector do TSS indexa para um *TSS Descriptor* na GDT.

O campo designado por DPL controla o acesso ao *TSS Descriptor*, de modo que uma rotina não terá acesso a mudança de Tarefa via *Task Gate* se o RPL do «selector» constante do segmento CS e o CPL da rotina não forem tanto ou mais priviligiados que o DPL da *Task Gate*.

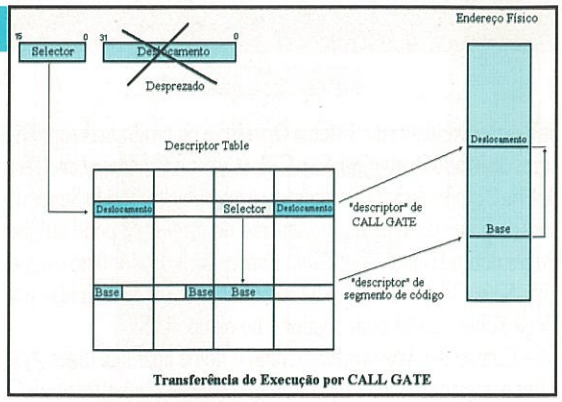
De notar que quando se utiliza uma *Task Gate* para mudar de Tarefa o DPL do *TSS Descriptor* de destino não é utilizado pelo processador.

E, como haviamos anunciado, vamos tratar agora das

CALL GATES

É possível também efectuar transferências para segmentos de código com níveis mais elevados de privilégio recorrendo a *Call Gates*. A figura anterior mostra-nos, além de outras, também a estrutura de um *Call Gate Descriptor* o qual pode residir quer na GDT, quer numa LDT, mas nunca na IDT.

O *Call Gate Descriptor* contém um «selector» para um *descriptor* de segmento de código numa das *Descriptor Tables*. Contudo, aqui será o valor do deslocamento contido no próprio *Call Gate Descriptor* que irá ser utilizado e não o valor de deslocamento determinado dentro do processo em execução, o qual será pura e simplesmente desprezado. Isto assegura que uma *Call Gate* fará a transferência de execução para um ponto perfeitamente bem determinado. Vejamos no esquema da figura seguinte como isto tudo se processa.



Transferência de Execução por CALL GATE

Durante a transferência de execução por *Call Gate* são verificados pelo processador os seguintes níveis de privilégio:

- CPL
- RPL
- DPL do *Call Gate Descriptor*.
- DPL do *descriptor* do segmento de código de destino.

O DPL do *Call Gate Descriptor* determina de que níveis de privilégio esta pode ser chamada. Quando as *Call Gates* são acedidas por instruções *Assembly* do tipo CALL vigoram as seguintes regras:

1) O DPL do *Call Gate Descriptor* será sempre maior ou igual que o CPL ou o RPL.
2) O DPL do *descriptor* do segmento de código de destino será sempre menor ou igual (maior ou idêntico nível de privilégio) que o CPL.

Quando se efectua uma transferência para uma rotina em nível mais elevado de privilégio através de uma *Call Gate*, o processador faz automáticamente o seguinte:

1) Actualiza o CPL
2) Muda de *stack*

O novo CPL corresponderá ao DPL do *descriptor* do segmento de código de destino.

Para mudar de *stacks* o processador encontra a informação necessária recorrendo ao *Task State Segment* tal como se escreveu atrás.

Os *Call Gate Descriptors* contêm ainda de especial um campo designado por «Contador» cujo valor informa o processador do número de *doublewords* que deverá copiar do velho para o novo *stack*.

E enfim, ficamos por aqui, a temática é por natureza muito difícil e alguns leitores terão já certamente pensado em desistir de nos continuar a ler, isto se outros entretanto não acabaram por adormecer com a luz do candeeiro acesa.

Demos uma panorâmica dos conhecimentos que consideramos mais cruciais. Um conhecimento efectivo obterá o leitor lendo algumas obras e estudando listagens de programas que versem estas matérias. As melhores obras são, sem dúvida, embora de leitura pouco fácil, os manuais de referência para programadores que são publicados pela Intel para os vários microprocessadores. Listagens de programas são efectivamente muito difíceis de encontrar, porém o programa VIRT86.EXE a ser publicado no próximo número da **Spooler** talvez contribua um pouco para preencher essa lacuna.

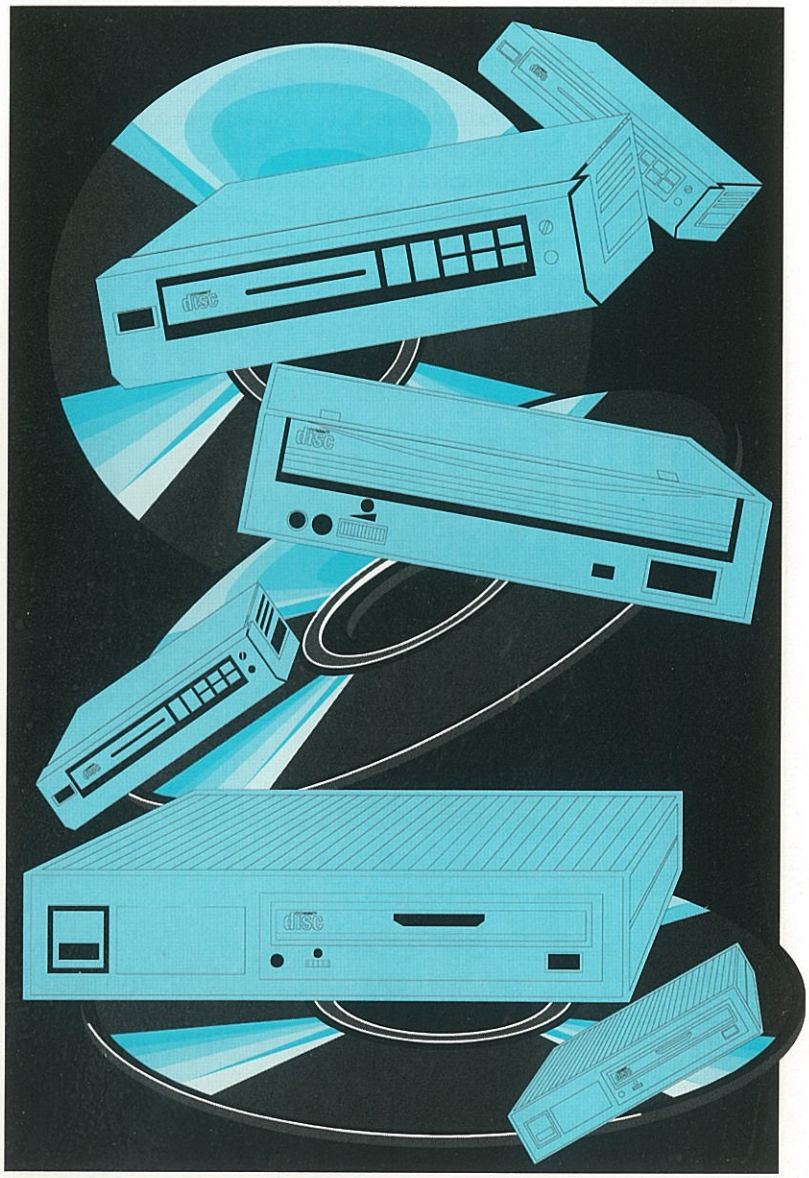
José Páscoa

CD-ROM PARA PC
A REVOLUÇÃO DO MULTIMEDIA (II)

Na continuação de uma série de artigos sobre a «revolução do multimedia» em PC, em que o primeiro (sobre placas de som) surgiu na **Spooler** Nº 25 (Outubro de 1993), desta vez irei debruçar-me sobre outro componente essencial num equipamento multimedia: a *drive* de CD-ROM.

O CD-ROM, como suporte de informação muito mais poderoso e fiável que os suportes magnéticos (disquetas, discos rígidos, *streamtapes*, etc) é em tudo idêntico aos vulgares CD musicais. Só que, ao ser lido num computador, pode armazenar programas, imagens, animações, enfim, tudo o que possa ser transformado em *bytes*. Como não pode ser regravado (uma vez lá armazenada informação, esta não pode ser alterada), comporta-se como uma ROM (*Read Only Memory*) e daí o seu nome – CD-ROM.

Devido à sua capacidade muito elevada (até 682 Mb), à sua fiabilidade e ao seu relativo baixo preço, o CD tornou-se por excelência o suporte ideal para quantidades muito grandes de informação, como a necessária para uma aplicação multimedia...



O PASSADO...

Apesar de os leitores de CD musicais existirem desde 1983, só no final dos anos 80 é que os leitores de CD-ROM para PC começaram a aparecer. No início eram dispendiosos e lentos – na realidade os leitores de CD musicais são relativamente lentos quando comparados com um disco rígido. Raramente acedem a uma faixa em menos de 600 milissegundos e a velocidade de transferência de dados é de 150 kilobytes por segundo.

Como as primeiras *drives* de CD-ROM eram quase idênticas aos leitores musicais, tinham as mesmas (baixas) *performances*. E como eram fabricadas em pequenas quantidades, custavam bastante acima de uma centena de contos.

Mas o seu potencial era óbvio e, com a passagem para os anos 90, começaram a vulgarizar-se a pouco e pouco, em particular no mercado norte-americano. Os preços foram baixando e as *performances* aumentando. Inicialmente, o tempo de acesso começou a cair para valores na ordem dos 400 ms, apesar da velocidade de transferência se manter. Começaram a aparecer as aplicações em CD-ROM, com especial destaque para enciclopédias multimedia que foram a primeira grande utilidade destas *drives*. No entanto, ainda se estava longe da interactividade, animação e qualidade gráfica e sonora desejável.

Finalmente, em 1992 começaram a surgir *drives* com tecnologias muito mais avançadas. A NEC lançou as primeiras *drives* com velocidades de transferência de 300 kb/s, chamadas de *double spin*, porque o CD rodava ao dobro da velocidade «normal». Tinham também tempos de acesso mais baixos, de 280 ms. Mas eram caríssimos.

Os leitores de baixo preço, ainda com velocidades de transferência de 150kb/s e tempos de acesso de 400 ms eram cada vez

mais vulgares e baratos. Graças a isto, o *software* em CD-ROM começou a aparecer em quantidade e qualidade. Não haviam apenas enciclopédias, mas também jogos de alta qualidade, aplicações interactivas, atlas e mesmo discos com 682 Mb de *shareware*...

... O PRESENTE...

Actualmente assiste-se a uma revolução neste campo. Existem já *drives* a preços incrivelmente baixos (menos de 30.000$00), como a Mitsumi LU005, com 350 ms de tempo médio de acesso e 150kb/s de velocidade de transferência de dados. No entanto, o *software* é cada vez mais exigente e os leitores *double-spin* começam a ser imprescindíveis. Estes têm baixado de preço e começam lentamente a vulgarizar-se. Existe já um Mitsumi *double-spin*, também com um preço relativamente baixo. Mas a grande estrela deste campo é o Toshiba XM-3401B, uma *drive* com um desempenho espectacular: 330 kb/s de velocidade de transferência de dados e 200 ms de tempo de acesso. Inclui uma *cache* interna (por *hardware*) de 256kb.

"Todas empresas de *software* em CD-ROM começam a trabalhar tendo como alvo o MPC nível 2."

Cada vez mais este tipo desempenho ser torna um mínimo. O *standard* MPC (Multimedia PC – a definição do que é um PC multimedia) foi recentemente alterado (e apelidado de MPC nível 2) para poder satisfazer as exigências do novo *software* mais exigente. Aqui temos um comparativo entre o *standard* MPC nível 1 e nível 2:

MPC Nível 1:

Processador 386SX a 16Mhz
2 Mb de RAM
30 Mb de disco rígido
CD-ROM de 150 kb/s
Placa de som de 8 bits
Placa gráfica VGA (8 bits)

MPC Nível 2:

Processador 486SX a 25Mhz
4 Mb de RAM (8 Mb recomendados)
160 Mb de disco rígido
CD-ROM *double-spin* (300 kb/s) compatível XA
Placa de som de 16 bits
Placa gráfica de 16 bits

(explicarei o que é a compatibilidade XA mais à frente)

Todas empresas de *software* em CD-ROM começam a trabalhar tendo como alvo o MPC nível 2. Assim, actualmente, um PC multimedia tem que satisfazer esses mínimos. Isto demonstra a evolução que houve em tão poucos anos. De um luxo, as *drives double-spin* tornaram-se um mínimo em apenas 2 anos...

Mas a evolução não acaba aqui. Os fabricantes levam a velocidade aos limites. Ainda em 1993 a Pioneer apresentou uma *drive* multi-CD (ou seja, que carrega não um mas 6 discos), a DRM604X, que é *quadra-spin*, ou seja, consegue velocidades de transferência de 600 kb/s... No entanto, o seu preço é incrivelmente alto, muito acima dos 200.000$00...

Para tentar travar o domínio do Toshiba XM-3401B (um *best-seller*, devido ao seu desempenho), a NEC apresentou os modelos 3X e 4X, que são respectivamente um *tri-spin* (450 kb/s) e *quadra-spin* (600 kb/s). O seu preço ainda não é conhecido, mas deverá ser elevado, especialmente o do modelo de 600 kb/s.

INTERFACES

Agora que cobrimos a parte da evolução do desempenho das drives, vamos às particularidades do seu *interface* com o computador. Tal como com os discos rígidos, é necessário existir um controlador / interface com o computador, para que estes possam comunicar. Inicialmente era utilizado o mesmo controlador das disquetas. No entanto, com o aumento do desempenho e o advento das placas de som, apareceram os interfaces «proprietários», em que cada *drive* tinha o seu controlador, que só funcionava com os leitores de uma marca (ou marcas) em específico. Exemplo disso é o controlador da Sound Blaster Pro (que só funciona com *drives* da Panasonic / Matsushita / Creative Labs), o da Sony e o da Mitsumi (em ambos os casos, algumas *drives* destas marcas só funcionam com um controlador em especial).

Estes controladores são extremamente baratos, mas além de limitarem a escolha, pois deixam-nos confinados a um leitor e controlador em específico, têm um desempenho bastante fraco. Felizmente que o SCSI (e o seu sucessor SCSI-2) começa, cada vez mais, a substituir os métodos «proprietários» e, actualmente, quase todas as *drives* com velocidades de transferência iguais ou superiores a 300 kb/s têm um interface SCSI. Exemplo disso é o Toshiba XM-3401B.

Assim, placas de som como a Pro AudioSpectrum 16, que já incluem um interface SCSI, podem servir de controladores, poupando-se a necessidade (e o dinheiro) de comprar um controlador à parte. Além disso, o SCSI permite desempenhos muito superiores a qualquer outro interface.

COMPATIBILIDADES

O problema é que não são só os controladores que não são todos iguais. Também há diferenças nos CD em si. Existem vários tipos de CD no mercado que podem ser lidos num leitor para PC:

CD-ROM (correcção de erros XA)
CD Audio (os musicais)
Photo CD
CD-I

Alguns formatos são definidos nos famosos *Yellow Book* e *Red Book*. Actualmente, todos os fabricantes tentam que os seus leitores sejam compatíveis com todos estes formatos.

Desde sempre que os leitores CD-ROM «tocam» os CD Audio, mas o XA e o Photo CD / CD-I são novidades. A correcção de erros XA é um dos mínimos necessários para satisfazer o MPC nível 2...

É evidente que, apesar da fiabilidade de um CD, este não é imune a riscos, poeira e dedadas. Assim, surgiu a correcção de erros XA, um *standard* para qualquer CD, com especial relevo para os Photo CD e CD-I (que usam o XA Modo 2, de que falaremos mais à frente). Esta permite verificar se a informação lida está correcta. É, hoje em dia, essencial em qualquer leitor. Além disso, ainda existe o pormenor do facto de os Photo CD

poderem ser gravados em várias sessões.

O Photo CD é um formato proposto pela Kodak e o seu princípio é simples: quando se tiram fotografias, em vez de serem reveladas para papel ou diapositivos, são colocadas num CD que pode armazenar várias dezenas de fotografias as quais podem ser mostradas no televisor com um leitor próprio. Este método, além de ser mais ecológico, é prático, poupa espaço e aumenta a durabilidade das fotos. Claro que, devido à capacidade de cada CD, para o encher podem ser necessários vários rolos, que são revelados em alturas diferentes. Quando já se tem um CD com fotos e se quer colocar outro rolo nesse CD, tem que se criar outra «sessão» de gravação de fotos.

O problema é que nem todos os leitores de CD-ROM para PC são capazes de ler a 2ª sessão e seguintes, devido à maneira como estas são gravadas no disco. Os leitores que conseguem ler a totalidade de um Photo CD, ou seja, todas as suas «sessões», têm que ser compatíveis XA Modo 2 e chamam-se *multisession* (multissessão).

O XA Modo 2 é uma extensão às especificações XA, e é um método de leitura especial, utilizado para os Photo CD. Todos os novos modelos de CD-ROM são *Photo CD multisession*. O Toshiba XM-3401B, mais uma vez, é um exemplo de uma *drive* que permite ler todos os 4 tipos de CD descritos acima, sendo por isso compatível XA, Modo 1 e 2.

SOFTWARE

Apesar de já ter falado superficialmente do *software* existente em CD-ROM, falta referir os *drivers* necessários para ler um CD. É que, devido à disparidade de controladores, bem como ao facto de ser um equipamento recente, o MS-DOS não traz suporte automático para CD-ROM, ao contrário do que acontece com as *drives* de disquetas e discos rígidos. Assim, sob o MS-DOS, é preciso carregar um *driver* do controlador e o MSCDEX (*Micro-Soft CD-ROM Extensions*).

Em relação ao primeiro, há que salientar que este é característico de cada placa controladora e é muito importante. É até uma das peças fundamentais da compatibilidade *Photo CD multisession*. Se o *driver* não for também compatível, o leitor não poderá ler a 2ª sessão e seguintes de um *Photo CD*.

Em relação ao MSCDEX, este é um TSR que vem com o MS-DOS 6.0 e 6.2 e que permite aceder ao CD-ROM como se tratasse de uma *drive* normal . Em relação ao MS-DOS 5.0 e anteriores, é preciso um MSCDEX à parte que deverá vir com o controlador do CD-ROM.

A placa de som PAS16, por exemplo, vem actualmente com o *driver* para CD-ROM SCSI próprio (compatível *Photo CD multisession*) e com um MSCDEX, apenas necessário para o MS-DOS 5.0 e anteriores.

Em relação ao Windows 3.1, este traz suporte para CD (o *Media Player* permite ler CD Audio), mas como este funciona sob o MS-DOS é à mesma necessário carregar o *driver* e o MSCDEX.

No OS/2 2.1 a coisa é bem diferente. O suporte integrado para multimedia é excelente e bastante mais simples que o do MS-DOS (mesmo que se use o Windows 3.1). O próprio sistema operativo pode ser comprado em CD-ROM, sendo mais barato (e muito mais fácil e rápido de instalar) que a versão que vem numas dezenas de disquetas. Mas não é só o OS/2 2.1 o único *software* comercial que é comercializado em CD além de em disquetas. Outros exemplos são o Corel DRAW! 4.0, o Windows NT 3.1 e muitos mais. Apesar de não tirarem partido das capacidades multimedia (texto, animação e som em simultâneo) dos CD-ROM, eles tiram partido da enorme capacidade do suporte e da velocidade dos leitores *double-spin*, ficando a sua instalação muito mais prática e rápida.

Em relação ao *software* verdadeiramente multimedia, existem milhares e milhares de títulos, desde jogos a programas educativos, com uma qualidade de sons e gráficos espantosa e com um nível de interactividade muito grande. Um bom exemplo disso é um dos mais famosos jogos em CD-ROM, o «The 7th Guest», que é um jogo de puzzles (extremamente divertido e um óptimo exercício mental) com níveis gráficos e sonoros espantosos, que no entanto só são possíveis com equipamento mínimo MPC nível 2. É autêntico filme interactivo, que já tem uma sequela.

No entanto, a evolução não pára e todos os dias sai mais *software* em CD-ROM com níveis de qualidade e interactividade cada vez melhores. Com o aumento constante das capacidades tanto do *hardware* como do *software*, o CD-ROM é o suporte de dados dos anos 90 e uma peça essencial de um equipamento multimedia.

Afonso Infante

BBSpooler

A BBS ao serviço dos leitores da Spooler

Todos os dias úteis: das 18.30 às 9.00 horas

Sábados, Domingos e Feriados: 24 horas

☎ 475 22 89

LIVROS

GUIAS DO UTILIZADOR

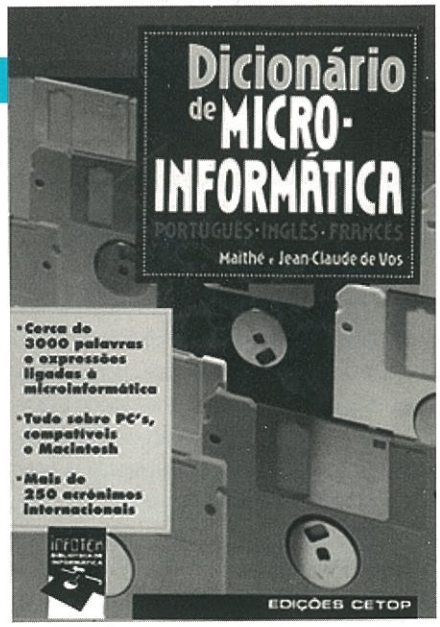
Estes guias são destinados aos utilizadores de computadores pessoais (PC ou Macintosh). Cada livro é consagrado a um programa específico ou a um sistema de exploração. A estrutura destes livros permite confirmar e solucionar qualquer dúvida sobre a sintaxe dos principais comandos, permitindo ao principiante descobrir rapidamente e iniciar a utilização do programa apresentado.

Títulos: WordBasic; Ventura Publisher 4.0; Guia para o Sistema 7 Macintosh; DR DOS 6; OS/2 Versão 2.0.
Idioma: Português
Editora e Distribuidora: Edições CETOP
Preço: 1.130$00 + 5% de IVA (por título).

O ESSENCIAL DO WORDPERFECT ATÉ À VERSÃO 5.1

O «Essencial do WordPerfect» é um guia simples e prático para quem necessite de aprender a utilizar este popular programa. Este guia fornece-lhe uma série de actividades práticas exemplificadas por numerosas figuras ilustrando o que aparece no ecrã do computador.

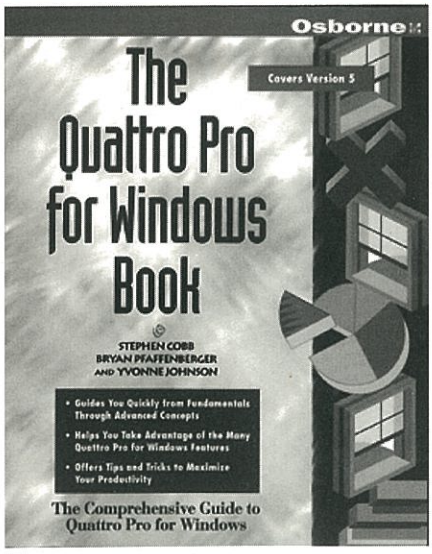
Título: O Essencial do WordPerfect até à Versão 5.1
Autoras: Ann Elms e Fenela Deards
Idioma: Tradução Portuguesa
Editora e Distribuidora: Edições CETOP
Preço: 1.810$00 + 5% de IVA



DICIONÁRIO DE MICROINFORMÁTICA

Um «rato» não é necessariamente apenas um mamífero roedor; uma célula pode designar outra coisa que um organismo vivo; uma maçã pode significar outra coisa que não um fruto... Este dicionário, com entradas em português, inglês e francês, facilitará a vida do leitor que, ao deparar-se com palavras ou expressões desconhecidas em qualquer uma das línguas, poderá saber rapidamente o significado das mesmas.

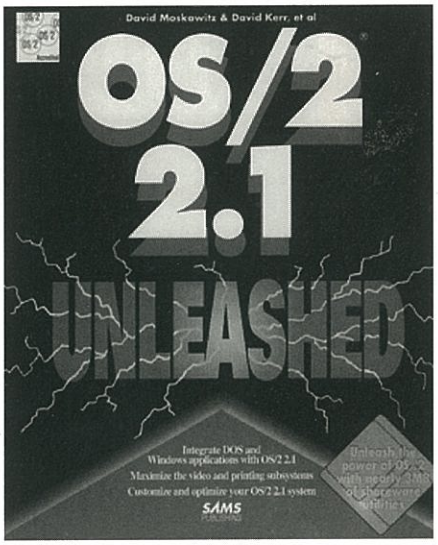
Título: Dicionário de Microinformática
Autores: Maithé e Jean-Claude de Vos
Idioma: Tradução Portuguesa
Editora e Distribuidora: Edições CETOP
Preço: 2.360$00 + 5% de IVA



THE QUATTRO PRO FOR WINDOWS BOOK

Um guia prático desde as bases até aos conceitos avançados, ajuda-o a tirar partido das muitas características do Quattro Pro Para Windows. Oferece-lhe, ainda, «dicas» e truques para aumentar a sua produtividade.

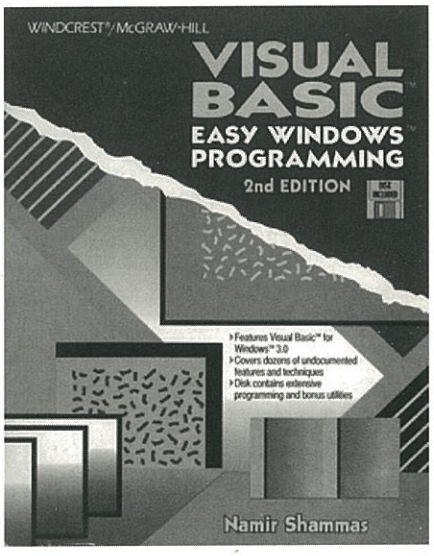
Título: The Quattro Pro for Windows
Autores: Stephen Cobb, Bryan Pfaffenberger e Yvonne Johnson
Idioma: Inglês
Editora e Distribuidora: MaGRAW-HILL de Portugal, Lda
Preço: 6.590$00 + 5% de IVA
ISBN: 881964-4



OS/2 2.1 UNLEASHED

Este livro oferece-lhe um inigualável nível de informação sobre as mais importantes características e funções do OS/2 2.1. Na disqueta incluída encontrará os melhores conselhos, truques e «dicas». O leitor aprenderá como interligar de forma eficiente o Windows, o DOS, O Workplace Shell e a linha de comando OS/2. Descubra características e truques não documentados. Torne-se num poderoso utilizador do OS/2. Optimize a sua utilização do Workplace Shell.

Título: OS/2 2.1 Unleashed
Autores: David Moskowitz e David Kerr, et al
Idioma: Inglês
Editora: SAMS Publishing
Distribuidora: LIDEL – Edições Técnicas, Lda
Preço: 10.100$00 + 5% de IVA
ISBN: 0-672-30240-3



VISUAL BASIC EASY WINDOWS PROGRAMMING

Uma ajuda prática e fácil para o guiar nas vantagens da Versão 3.0 do Visual Basic, permitindo-lhe escrever aplicações com qualidade profissional. Cobre, entre outros os seguintes tópicos:

- Criação e utilização de menus
- Trabalhar com formatos múltiplos
- Direccionar a saída para formatos, caixas de imagem e impressora
- Utilização das capacidades de manipulação do rato

Título: Visual Basic Easy Windows Programming
Autor: Namir Shammas
Idioma: Inglês
Editora e Distribuidora: MaGRAW-HILL de Portugal, Lda
Preço: 9.500$00 + 5% de IVA
ISBN: 08306444504

IOMEGA BERNOULLI ARMAZENAGEM AMOVÍVEL DE DADOS

Bernoulli MultiDisk™ 150

Fácil como um floppy com as prestações de um hard disk

- 150 MB (300 MB comprimido - soft incluído)
- 18 ms de tempo de acesso efectivo
- Robusta Tecnologia Bernoulli
- Multi-Disk 150, 105, 65, 35 MB
  A capacidade certa para cada tarefa
- Compatível leitura-escrita com Bernoulli 90 MB
- Compatível leitura com Bernoulli 44 MB

Características - Vantagens

- Prestações a nível de Hard Disk
- Maior capacidade para satisfazer as necessidades crescentes e Multi-Media, DTP, Gráficos, Apresentações, CAD e Programação
- Ideal para Windows e Quicktime quando os utentes necessitam de mais espaço
- Somente a Bernoulli resiste a aterragem das cabeças e somente os discos Bernoulli suportam choques de 1000 G (queda de $2^1/_2$ metros sobre piso de betão)
- Soluções prontas a usar para PC e MAC
- Soluções universais - conectável com PC, MAC, Workstation e Portátil

Modelos disponíveis

- Insider (Encastrável em espaço $5^1/_4$") MultiDisk 150 para PC e MAC
- Transportável MultiDisk 150
- Dual MultiDisk 150
- PC Powered 150
- MacTransportável 150

Iomega® 21 MB Floptica®

Numa só drive utiliza a Floppy Normal $3^1/_2$" e a Floptica 21 MB

- 21 MB de capacidade numa diskette de baixo custo
- Escrita e Leitura de floppies normais
- Até 3 vezes mais rápida do que uma floppy normal
- HOT (Holographic Optical Tracking) permite maiores prestações e fiabilidade do que outras flopticas
- Novas diskettes gravadas a LASER da IOMEGA - Todas as diskettes são diskettes padrão

Características - Vantagens

- Soluções para PC e MAC
- 14 vezes a capacidade duma diskette de alta densidade
- 3 vezes mais rápidas do que as floppies
- Forma mais económica de aumentar a capacidade
- Floptica mais rápida do mercado
- A concepção Laser Holográfica e mais simples, rápida e fiável
- Sistemas completos duma só marca - drive, adaptador, software e suportes gravados a LASER
- Diskettes disponíveis em embalagens individuais ou de 5

Modelos disponíveis

- Floptica Insider (Encastrar para $3^1/_2$")
- Floptica PC Powered (Externa)
- Subsistema Floptica MAC
- Floptica Parallel Port (liga por cabo a porta paralela)

Iomega® TAPE 250

A streamer que faz 5 vezes mais pelo mesmo preço

- Cabe no espaço de 1 floppy 31/2"
- Modelo especial para ligar a porta paralela
- A única Streamer que também lê cassetes QIC e IRWIN
- Software Central Point com suporte de WINDOWS incluído

Características - Vantagens

- A única drive de 1" compatível com IRWIN e QIC
- Software WINDOWS incluído
- O modelo para ligar a Porta Paralela pode usar um cabo até 5 metros de comprimento
- Compatível com praticamente todo o Software elaborado para QIC

Modelos disponíveis

- TAPE 250 Insider (Encastrável $3^1/_2$")
- TAPE 250 Insider Half-Height (Encastrável $5^1/_4$")
- TAPE 250 PARALLEL PORT (liga por cabo a porta paralela)

Iomega® LaserSafe Plus

Drive magneto--optico regravável

- Capacidade 1.3 Gigabytes
- Tempo de busca 18 MS
- Discos seguros em arquivo por 30 anos
- Comutador de modo de operação permite operação com suporte fixo ou amovível
- Compatível com norma ISO - lê e grava discos 650 MB, MO

Características - Vantagens

- Drive Magneto-Optica de maior capacidade
- Prestações próximas das dos Hard Disks
- Soluções para PC, PS2 e Workstations

movicontrol

Sistemas de Accionamento e Automação, Lda

LISBOA
QUINTA DO MARCHANTE, 14
PRIOR VELHO
2685 SACAVÉM
☎ (01)942 03 05 PPCA
Telefax: (01) 941 35 00
Telex: 62492 MOVILX P

PORTO
PRC. PEDRA VERDE, 217 e 222
4100 PORTO
☎ (02) 610 41 76/7/8
Telefax: (02) 610 44 75
Telex: 29192 MOVIPO P

MEMBROS
SUBSCRITORES
ASSOCIAÇÃO PORTUGUESA
PARA A QUALIDADE

SOCIEDADE POR QUOTAS
MATRÍCULA 6115 C.R.C. - LOURES
PESSOA COLECTIVA N.º 501 794 638
CAPITAL REALIZADO: 80 000 000$00

Marque no Cupão RSF o Nº 79

Consumíveis para Informática

Somos especialistas em:

- **Duplicação de Software**
  (Possuimos equipamentos específicos)
- Diskettes personalizadas
- Fitas para impressoras (originais ou compatíveis)
- Tinteiros para impressoras de Jacto de Tinta
- Tapetes e suportes para «ratos»
- Suportes para documentos no monitor
- Caixas arquivadoras para diskettes
- Etiquetas em contínuo ou folhas soltas para laser
- Rolos para Fax
- Toners para Lasers
- Toners para fotocopiadoras
- Capas para CPU, teclados e impressoras
- Consumíveis para **Hewlett Packard**
- Filtros **Polaroid** para ecrãs
- Data, Tape Cartridges
- Bandas Magnéticas

LM ELECTRÓNICA DE CONSUMO, LDA.

Rua Barão Sabrosa, nº 31 - C – 1900 LISBOA
Telef.: 814 63 74 – Fax: 814 17 96